



**DISCOVERING THE MERIT OF THE WAVELET TRANSFORM FOR OBJECT  
CLASSIFICATION**

THESIS

Matthew D. Eyster, Captain, USAF

AFIT/GE/ENG/04-09

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

---

---

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GE/ENG/04-09

**DISCOVERING THE MERIT OF THE WAVELET TRANSFORM FOR OBJECT  
CLASSIFICATION**

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the  
Degree of Master of Science in Electrical Engineering

Matthew D. Eyster, BSEE

Captain, USAF

March 2004

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

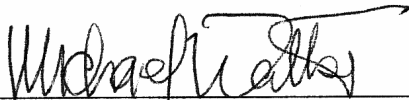
AFIT/GE/ENG/04-09

**DISCOVERING THE MERIT OF THE WAVELET TRANSFORM FOR OBJECT  
CLASSIFICATION**

Matthew D. Eyster, BSEE

Captain, USAF

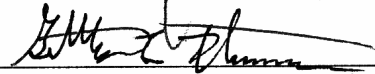
Approved:



Michael L. Talbert, Lt Col, USAF PhD (Thesis Advisor)

09 Mar 04

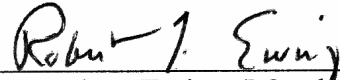
Date



Dr Gilbert Peterson (Member)

9 MAR 04

Date



Dr Robert Ewing (Member)

10 Mar 04

Date

## **Acknowledgments**

First and foremost, I would like to say thank you to my wife for supporting me through the AFIT experience. Thank you for always supporting me and putting up with those long hours. Also, I would like to say thank you to my children for hanging in there when Dad was up late working.

Lt. Col. Talbert, thank you for your encouragement when I felt like the tunnel would never end. Thank you for nudging me in the direction that I needed to go and giving me a fresh perspective when I needed it most.

I would also like to acknowledge the others who helped me along this road. Dr. Ewing, you provided the initial direction of this research. It has proved to be an interesting topic area. Dr. Peterson, I appreciate how you always allowed me to dash into your office and question you about neural networks.

Matthew D. Eyster

## Table of Contents

	Page
Acknowledgments.....	iv
Table of Contents.....	v
List of Figures.....	ix
List of Tables .....	xiii
Abstract.....	xiv
I. Introduction .....	1
1.1 Research Focus.....	2
1.1.1 Problem Statement.....	2
1.1.2 Objective and Approach.....	3
1.1.3 Scope.....	4
1.2 Sponsor Acknowledgement.....	4
1.3 Document Overview.....	4
II. Literature Review.....	5
2.1 Neural Networks.....	5
2.1.1 Overview and Advantages.....	5
2.1.2 Biological Equivalence.....	7
2.1.3 Types of Neural Networks.....	9
2.1.4 Physical Architecture.....	10
2.1.5 Learning.....	13
2.1.6 Training of Neural Networks.....	18
2.1.7 Network Models.....	24

2.1.8 Pattern Recognition.....	27
2.1.9 Hardware Implementation. ....	28
2.2 Wavelet Transform.....	29
2.2.1 Fourier Transform.....	29
2.2.2 Wavelet Transform. ....	31
2.3 Object Recognition.....	38
2.3.1 Difficulties. ....	39
2.3.2 Approach.....	41
2.4 Summary.....	46
III. Methodology.....	48
3.1 Approach .....	48
3.2 System boundaries and Overview .....	50
3.3 System Services.....	51
3.4 Performance Metrics .....	51
3.5 Parameters .....	52
3.5.1 System Parameters. ....	52
3.5.2 Workload Parameters.....	54
3.6 Factors .....	56
3.6.1 System Factors. ....	56
3.6.2 Workload Factors.....	59
3.7 Approach in Detail .....	63
3.7.1 Creation of an Image Database.....	63
3.7.2 Wavelet Transformation of Each Image.....	65
3.7.3 Scaling Function. ....	66

3.7.4	Determine Optimal Neural Network Topology and Data Transform.....	68
3.7.5	Results of Trial Networks.....	70
3.7.6	Testing of the Trained Neural Networks.....	71
3.7.7	Create a Network Composed of the Individual Subband Networks.....	72
3.7.8	Testing of the Final Network.....	73
3.7.9	Additional Verification Tests.....	73
3.8	Evaluation Technique.....	74
3.9	Experimental Design.....	74
3.10	Analyze and Interpret Results.....	76
3.11	Summary.....	76
IV.	Analysis and Results.....	78
4.1	Level 3 (Lowest Resolution).....	79
4.3	Level 2.....	88
4.3	Level 1 (Highest Resolution).....	92
4.4	Multi-Level Networks.....	96
4.5	Verification Tests.....	99
4.6	Summary.....	102
V.	Conclusions and Recommendations.....	103
5.1	Summary.....	103
5.1.1	The Wavelet Transform as a Pre-Processor.....	103
5.1.2	Scaling Function.....	104
5.1.3	Network Topology.....	104
5.1.4	Multi-resolution Wavelet Transform.....	105
5.2	Recommendations for Future Research.....	105



5.2.1 Wire-frame Models.....	105
5.2.2 Additional Training.....	106
5.2.3 Data Transform.....	106
5.2.4 Scale, Rotation, and Translational Invariant Transform.....	106
5.2.5 Rotation about Axes.....	107
5.3 Final Thoughts.....	107
Appendix A: Results for Level 3 .....	108
Appendix B: Results for Level 2 .....	112
Appendix C: Results for Level 1 .....	121
Appendix D: Results for Training and Testing with Photo Sets.....	130
Appendix E: MATLAB Code for Computing Trained Epoch.....	134
Bibliography .....	137
Vita .....	140

## List of Figures

	Page
Figure 2.1. Biological neurons.....	7
Figure 2.2. Neural network with a single hidden layer, a partially connected layer, and a fully connected layer. ....	12
Figure 2.3. Examples of mean square error for an undertrained and overtrained neural network.....	23
Figure 2.6. Histogram display of the distribution of wavelet coefficients.....	37
Figure 2.7. Histogram display of the distribution of wavelet coefficients for LL subband and across all three scales.....	37
Figure 2.8. Parallel and orthogonal lines in an image.....	42
Figure 2.9. Vanishing point in an image.....	42
Figure 2.10. The line appears continuous due to applied domain knowledge. ....	45
Figure 3.1. Simplified model of the image correlation system.....	50
Figure 3.2. Smooth textured CAD models of the objects used in this research. ....	60
Figure 3.3. Photos of the real-life models.....	60
Figure 3.4. Textures used to surface CAD models. ....	62
Figure 3.5. Creation of the pattern sets.....	68
Figure 3.6. Determination of a neural network's "trained epoch" (the epoch at which the network is fully trained).....	71
Figure 3.7. Large correlating network connecting the subband networks.....	73
Figure 4.1. Levels and subbands of the wavelet decomposition.....	78
Figure 4.2. Level 3 variability chart for the CAD set success rate. ....	82
Figure 4.3. Level 3, HH subband trained epoch. ....	83

Figure 4.4. Large correlating network connecting the subband networks for a level.....	84
Figure 4.5. Results of the CAD testing set on the level 3 “glue” fully connected network with hidden nodes. ....	85
Figure 4.6. Results of the CAD testing set on the level 3 “glue” fully connected network without hidden nodes. ....	86
Figure 4.7. Results of the CAD testing set on the level 3 correlating network.....	87
Figure 4.8. Level 2 variability chart for the CAD set success rate and trained epoch. ....	90
Figure 4.9. Level 1 variability chart for the CAD set success rate and trained epoch. ....	94
Figure 4.10. Success rates for subband and level networks tested on the CAD testing set. ....	97
Figure 4.11. Success rates for subband and level networks tested on the good photo set. ....	98
Figure 4.12. Results of processing unmodified spatial images on a single neural network.....	101
Figure A.1. Level 3 variability chart for the CAD set trained epoch.....	108
Figure A.2. Level 3 variability chart for the CAD training set mean square error. ....	109
Figure A.3. Level 3 variability chart for the CAD testing set mean square error. ....	110
Figure A.4. Level 3, HL subband trained epoch.....	111
Figure A.5. Level 3, LH subband trained epoch.....	111
Figure B.1. Level 2 variability chart for the CAD set success rate. ....	112
Figure B.2. Level 2 variability chart for the CAD set trained epoch.....	113

Figure B.3. Level 2 variability chart for the CAD testing set mean square error. ....	114
Figure B.4. Level 2 variability chart for the CAD training set for success rate.....	115
Figure B.5. Level 2, HH subband trained epoch.....	116
Figure B.6. Level 2, HL subband trained epoch.....	116
Figure B.7. Level 2, LH subband trained epoch.....	117
Figure B.8. Results of the CAD testing set on the level 2 “glue” fully connected network with hidden nodes.....	117
Figure B.9. Results of the CAD testing set on the level 2 “glue” fully connected network without hidden nodes.....	118
Figure B.10. Results of the CAD testing set on the level 2 correlating network.....	118
Figure B.11. Results of the CAD testing set on the level 2 and 3 “glue” fully connected network with hidden nodes.....	119
Figure B.12. Results of the CAD testing set on the level 2 and 3 “glue” fully connected network without hidden nodes.....	119
Figure B.13. Results of the CAD testing set on the level 2 and 3 correlating network.....	120
Figure C.1. Level 1 variability chart for the CAD set success rate. ....	121
Figure C.2. Level 1 variability chart for the CAD set trained epoch.....	122
Figure C.3. Level 1 variability chart for the CAD testing set mean square error. ....	123
Figure C.4. Level 1 variability chart for the CAD training set mean square error. ....	124
Figure C.5. Level 1, HH subband trained epoch.....	125
Figure C.6. Level 1, HL subband trained epoch.....	125

Figure C.7. Level 1, LH subband trained epoch.....	126
Figure C.8. Results of the CAD testing set on the level 1 “glue” fully connected network with hidden nodes.....	126
Figure C.9. Results of the CAD testing set on the level 1 “glue” fully connected network without hidden nodes.....	127
Figure C.10. Results of the CAD testing set on the level 1 correlating network.....	127
Figure C.11. Results of the CAD testing set on the level 1, 2, and 3 “glue” fully connected network with hidden nodes.....	128
Figure C.12. Results of the CAD testing set on the level 1, 2, and 3 “glue” fully connected network without hidden nodes.....	128
Figure C.13. Results of the CAD testing set on the level 1, 2, and 3 correlating network.....	129
Figure D.1. Variability chart for the photo set success rate.....	130
Figure D.2. Variability chart for the photo set trained epoch.....	131
Figure D.3. Variability chart for the photo testing set mean square error.....	132
Figure D.4. Variability chart for the photo training set mean square error.....	133

## List of Tables

	Page
Table 3.1. Example data collection table for success rate for level 3, subband 3HH. ....	77
Table 4.1. Level 3 results for each factor.....	80
Table 4.3. Subband and level network success rates (%) for level 3.....	88
Table 4.4. Level 2 results for each factor.....	89
Table 4.5. Results of level 2 ran with random seeds. ....	91
Table 4.6. Subband and level network success rates (%) for level 2.....	91
Table 4.7. Level 1 Results for each factor. ....	93
Table 4.8. Results of level 1 subband networks ran with random seeds. ....	95
Table 4.9. Subband networks' and level networks' success rates (%) for level 1.....	95
Table 4.10. Overall results for all networks.....	98
Table 4.11. Success rates of photo training vs. CAD model training.....	99
Table 4.12. Computation requirements of neural networks used in this research. ....	100

## **Abstract**

Vision is the primary sense by which most biological systems collect information about their environment. Computer vision is a branch of artificial intelligence concerned with endowing machines with the ability to understand images. Object recognition is a key part of machine vision with far reaching benefits ranging from target recognition, surveillance systems, to automation systems.

Extraction of salient features from an image is one of the key steps in object recognition. Typically, geometric primitives are extracted from an image using local analysis. However, the wavelet transform provides a global approach with good locality. Additionally, the directional and multiresolution properties may be exploited as a pre-processor to a neural network.

This thesis examines the benefits of the wavelet transform as a pre-processor to a neural network for object recognition. Scaling of the wavelet coefficients and different neural network topologies are investigated. The system developed in this research is not intended to be critiqued on its classification performance. It only successfully classifies about 20% of the photographed models, however more important is the determination of the benefits of the wavelet transform, the effects of the various post-wavelet scaling functions, and the best neural network topology for this research. This is done by analyzing the system's performance on CAD models.

# **DISCOVERING THE MERIT OF THE WAVELET TRANSFORM FOR OBJECT CLASSIFICATION**

## **I. Introduction**

Three dimensional object recognition is a very difficult problem. Biological systems make this task seem easy, yet it proves surprisingly challenging for machines. Many different approaches have been taken with varied levels of success. Human vision has little trouble isolating the object of interest from an image, comparing that object to another, and deciding whether it is the same object. This is a common task performed thousands of times daily with never a second thought. However, for machines this task represents multiple problems that involve large amounts of computational resources.

The first step in object recognition is isolating the objects of interest from an image. The human visual system (HVS) can focus on an object of interest to extract further details. Noise and distortion pose little problems for the HVS.

Once the object is isolated, the human eye depends on several cues to determine the three dimensional geometry. Texture, shading, contours, and motion are all cues used by the HVS to derive geometry information.



Classifying the object into pre-arranged classes is another significant problem. Classification compares an object to a pre-existing model rather than having to form a new class as in the more difficult problem of object recognition.

These problems are approached many different ways with varied levels of success. In spite of the many hurdles, the benefits offered by machine vision easily make the goal worth pursuing. Target tracking, surveillance, and automated systems are some of the rewards of machine vision.

## **1.1 Research Focus**

This research focuses on determining the benefits of using a wavelet transform as a data pre-processor to a neural network for object classification. The wavelet transform yields directionally sensitive, multi-resolution subbands that may be successfully exploited by neural networks for object classification. Additionally, several rescaling functions of the wavelet coefficients are explored to further optimize the performance of the neural network. A number of neural network topologies are also investigated to determine which proves the most successful for object classification.

### **1.1.1 Problem Statement.**

Image recognition is a complicated problem. Therefore, the somewhat less difficult problem of image classification is engaged. The system developed in this research is not intended to solve the image recognition problem, but only provide some insight into the possible benefits of the wavelet transform. Specifically, the possible

benefits of using the wavelet transform as a neural network pre-processor are explored in this research.

### **1.1.2 Objective and Approach.**

The main objective of this research is to discover if the wavelet transformation can be used as a pre-processor to a neural network for successful object classification. Further goals are to examine what data transformations, if any, increase the success rate and what neural network topologies prove the most successful. The final goal is to determine if the multiresolution aspect of the wavelet transform is usable to increase the success rate by processing each wavelet subband independently and then correlating the results.

To meet these goals, the wavelet transform of an image is computed, optionally processed with a data scaling function, and finally processed with various neural network topologies. The system ultimately produces a success rate based on how many of the presented images are correctly classified. Ideally, the objects should successfully correlate regardless of the object's elevation angle, rotation, and scale. Examination of the results of using the directionally sensitive wavelet subbands as inputs to neural networks should provides some insight into the worth of wavelet transforms as feature extractors.

### **1.1.3 Scope.**

Image recognition is a complex problem. This research engages the subset problem of image classification. This should still identify the possible benefits of the wavelet transform. Furthermore, the data set is constrained to a limited number of objects. The data images used in this research are created so that the system is not further hampered with the task of isolating the object of interest or challenged by excessive noise.

## **1.2 Sponsor Acknowledgement**

This research was sponsored by the Air Force Research Laboratory (AFRL), Embedded Information Systems Engineering Branch (IFTA). IFTA develops information system technologies that enable command and control of both current and next-generation aerospace weapon systems.

## **1.3 Document Overview**

This thesis reports research performed while exploring the wavelet transform as a pre-processor to a neural network for object classification. Chapter II provides an introduction to neural networks, the wavelet transform, and object recognition. Chapter III outlines the methodology used in this research and the experiments performed. Chapter IV provides results, analysis, and conclusions. Chapter V concludes with a final summary and future recommendations.

## II. Literature Review

This chapter provides some background information about the subject areas involved in this research. Throughout Chapter 2, certain sentences are in bold print to indicate a decision directly affecting the setup of this research. Section 2.1 covers the principles of neural networks and the reasons for choosing the particular neural network model used in this research. These reasons are presented in a concise list in the chapter summary in Section 2.4. Section 2.2 briefly discusses the wavelet transform, especially as applied to imagery. Section 2.3 discusses object recognition and classification. Additionally, decisions regarding the data set are emphasized with bold print.

### 2.1 Neural Networks

Neural networks (NNs) form an artificial intelligence system that is modeled after the nervous system. Neural networks are used to solve complex problems and problems that are difficult to formulate by using a nonlinear model.

#### 2.1.1 Overview and Advantages.

Neural networks have several properties that make them very useful. In brief, these properties are [Hayk94].

- *Non-linearity.* Neural networks are inherently nonlinear due to the non-linearity of each of the neurons composing the network.
- *Input-output mapping.* Neural networks often use a supervised learning system in which thresholds and weights of the NN are adjusted to meet the desired output

response. This input-output mapping is similar to nonparametric statistical inference (model-free estimation), but does not require a probabilistic distribution model.

- *Adaptivity.* NNs can adapt to their environment by changing thresholds and weights and may perform better in a changing environment. However, this adaptability comes at the cost of stability.
- *Evidential Response.* Besides merely providing a selection as in the case of a classifying neural network, the confidence with which the decision is made can also be determined. This information can be used to improve the classification performance.
- *Contextual information.* The structure and activation state of the network stores information. Changes to neurons can globally affect other neurons.
- *Fault Tolerance.* Since information is distributed throughout the network, the neural network has some inherent fault tolerance and tends to fail gracefully as damage occurs to neurons.
- *VLSI implementability.* Neural networks are massively parallel. This parallelism lends itself to implementation using VLSI hardware.
- *Uniformity of analysis and design.* Neural networks share the same notation regardless of application; therefore it is possible to share theories and learning algorithms. Additionally, it is also possible to join networks together.
- *Neurobiological analogy.* The human brain is the best example of a fast and powerful fault-tolerant, massively parallel architecture.

### 2.1.2 Biological Equivalence.

The nervous system is composed of a large network of neurons. Each neuron has three components: the dendrites, the cell body, and the axons (nerve fibers). Dendrites are transmitters and axons are receivers. The nervous system affects the conductivity of the synapse, the area between the dendrites and the axons. Synapses are excitatory or inhibitory. Neurons interact when the axons and dendrites are close enough together and the axons receive a strong enough signal from one or more dendrites. In animal nervous systems, thousands of neurons are close enough together to interact, forming a complicated network.

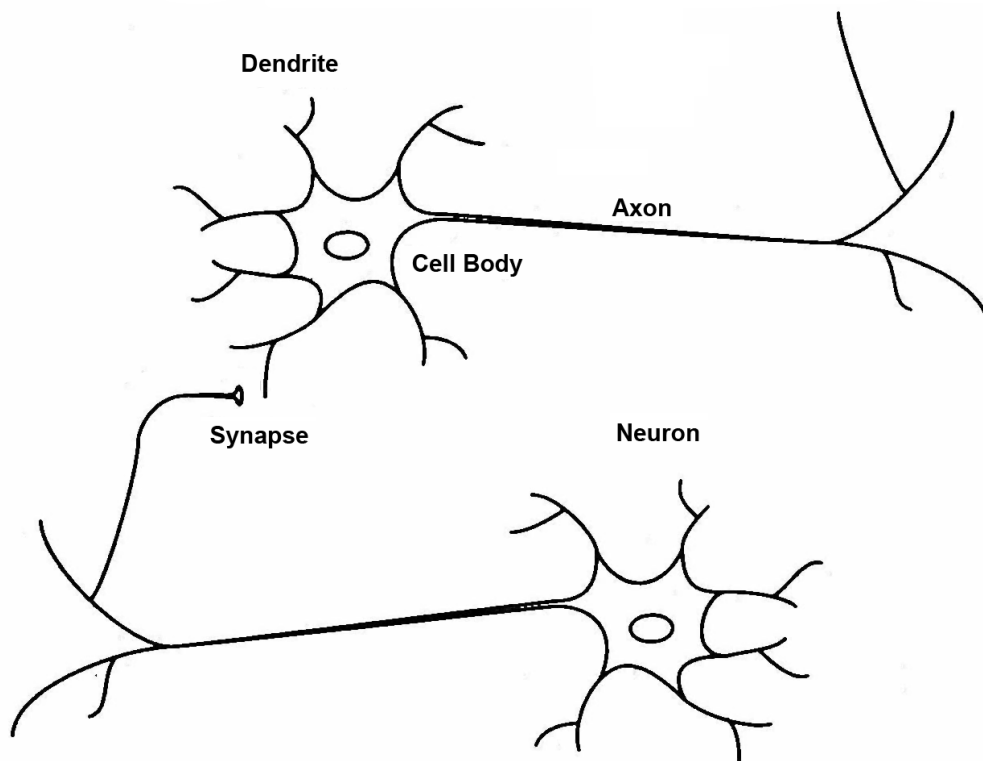


Figure 2.1. Biological neurons.

When a neuron receives a signal from other neurons, it causes the receiving neuron's voltage to increase through an electro-chemical process. Neurons have multiple

inputs which are weighted differently. When the cumulative received voltage exceeds a certain threshold, the neuron fires sending a signal to another neuron, which may in turn be stimulated enough to fire. Neurons have a certain dynamic range from no response to a full response and exhibit a nonlinear behavior. This behavior is difficult to analyze mathematically, yet is essential for the effectiveness of neural networks.

Artificial neural networks (ANN) operate in the same manner as biological nervous systems. When the combination of inputs surpasses the threshold, a neuron fires a weighted signal to other neurons. This activation function can be a linear threshold function (all-or-nothing), a piecewise-linear function, or a sigmoid function depending on the neural network model used. A single neuron cannot perform basic Boolean functions such as an exclusive OR. However, just three neurons are capable of solving this problem. This is described in more detail in Section 2.1.8. Larger networks can solve more difficult problems and can compute any Boolean function. Binary threshold logic units with unit time delays can realize any finite state automata and a sufficiently large finite state machine can compute any finite algorithm. Therefore, in theory, neural networks using threshold neurons should be capable of computing any computable function.

Neural networks depend on many simple processors with few processing steps while conventional computers have few complex processors that perform many computational steps. Neural networks employ a distributed processing paradigm that allows for graceful degradation while conventional computers rely on symbolic processing that suffer from catastrophic failure if any component should fail. In short, biological neural networks rely on many parallel, relatively slow operations (10 billion

neurons and 60 trillion synapses that operate in the millisecond range) whereas conventional computers are usually single threaded yet very fast (nanosecond range).

Neural networks provide a form of artificial intelligence and, individually, each node is easily computable. Computers can realize complex neural networks, provided that the computers have enough memory. However, neural networks have some major problems that limit their performance. For instance, it is difficult to determine the following properties:

1. Which neurons should be connected
2. The weights for each of the edges (the strength of firing between neurons)
3. The time delta allowed for signal arrivals to add together

Even though the back-propagation training method partially solves the second problem, it is still slow and imperfect. Additionally, new problems arise such as stability and convergence. The weights and thresholds of the neurons must be adjusted when the neural network's output does not match the desired output. The rules that determine how these changes are made are called the learning algorithm and are often the most difficult aspect of a neural network. Yet another failing of neural networks is that the computational complexity increases exponentially with the size of the network.

### **2.1.3 Types of Neural Networks.**

Neural networks can employ a variety of network architectures, algorithms, and training methods. The architecture of the neural network is closely linked with the



learning algorithm used. The major characteristics of neural networks are discussed in this section.

#### 2.1.4 Physical Architecture.

The architecture is the way in which the neurons interconnect in the neural network. Architecture encompasses the number of layers, the connections between the layers, feedback loops, and the firing model. The firing model of the neurons is either probabilistic or deterministic. Probabilistic neurons fire with a certain probability when stimulated while deterministic neurons always fire when the inputs exceed a pre-determined threshold. Additionally, the activation function (firing model) uses a threshold function, a piecewise-linear function, or a sigmoid function. A common activation function is the sigmoidal nonlinearity defined by the logistic function:

$$y_j = \frac{1}{1 + e^{-v_j}} \quad (2.1)$$

where  $y_j$  is the output of neuron  $j$ , and  $v_j$  is the total internal activity (sum of all inputs) of that neuron. **The logistic activation function is used in this research due to its desirable nonlinearity, biological equivalence, and the fact that it is “by far the most common form of activation function used in the construction of artificial neural networks,” [Hayk94].**

Neural networks are organized in layers of one or more neurons in each layer. The number of neurons may differ in each layer. In the most basic model, there is an input layer connected directly to an output layer of computational nodes. Hidden layers often exist between the input and output layers. Hidden layers allow the NN to extract

higher-order statistics and achieve a global perspective instead of only achieving a local perspective. Local features are extracted in the first hidden layer and global features are extracted in the second hidden layer. A single hidden layer is not enough to solve all inverse problems such as inverse kinematics (dynamics) and control problems. However, multilayer feedforward networks with two hidden layers are sufficient for solving these type of problems because they can approximate the required functions. Neurons tend to interact with each other globally when using a hidden layer. However, this “interaction makes it difficult to improve the approximation at one point without worsening it at some other point,” [Hayk94]. In other words, the network takes longer to train and is inherently less stable. For these reasons, **a four layer network using two hidden layers is used for this research.**

Neural networks interconnected in a feedforward manner or a feedback manner. A feedforward scheme passes the outputs from each layer into the next layer as inputs without feedback. Alternately, a recurrent neural network uses a feedback scheme that can have one or more feedback loops. Recurrent networks employ unit-delay elements which result in a nonlinear dynamical behavior. Recurrent networks such as the Hopfield network, the Boltzmann machine and the MFT machine require time to settle to equilibrium condition and therefore may be excessively slow. For this reason, **recurrent networks are avoided and a feedforward network is used for this research.**

Full or partial connections exist between layers. In a fully connected network, every neuron in a layer is connected to every neuron in the next layer. A partially connected network does not connect a neuron to every neuron in the next layer. Figure 2.2 shows an example of a neural network containing the different elements discussed.

The input layer is partially connected to a hidden layer. The hidden layer is fully connected to the output layer.

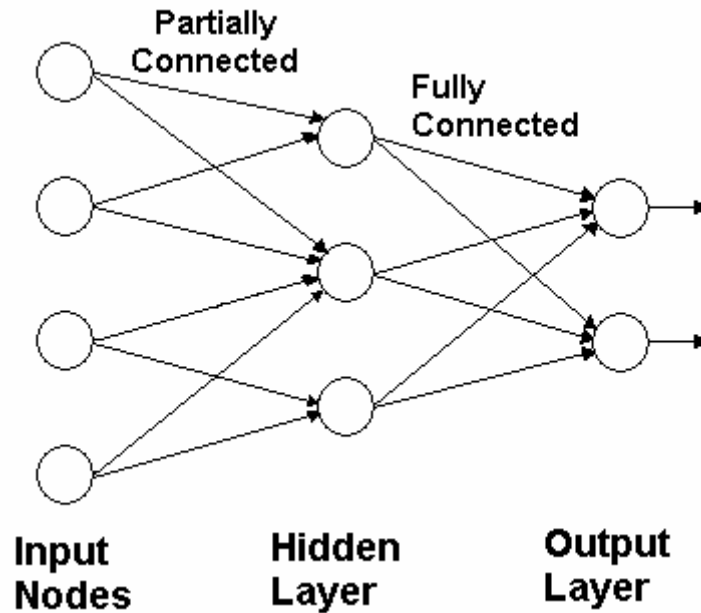


Figure 2.2. Neural network with a single hidden layer, a partially connected layer, and a fully connected layer.

The feedforward procedure operates by calculating the product of each neuron's output and its weight connection to each of the nodes in the next layer. The net sum of all inputs is calculated for each node. This process must repeat for every layer of nodes. As an example, consider a network used in research consisting of 256 nodes in the first layer, 128 nodes in the first hidden layer, 32 nodes in the second hidden layer, and 8 nodes in the output. This network would take approximately 37,120 operations to process each input pattern. This figure is calculated as follows:

$$(Inputs \times Hidden1) + (Hidden1 \times Hidden2) + (Hidden2 \times Outputs) = Operations \quad (2.2)$$

When applied to this example network, the number of operations required is

$$(256 \times 128) + (128 \times 32) + (32 \times 8) = 37,120 \text{ operations} \quad (2.3)$$

The multilayer perceptron (MLP) and radial basis function (RBF) networks are the two most commonly used network architectures [Sar196]. Both networks, in theory, can solve any solvable function, but “the MLP architecture is the most popular one in practical applications,” [Hayk94]. The multilayer perceptron utilizing the back-propagation training method has solved difficult and diverse problems. The hidden layers, high degree of connectivity, nonlinear nature of the output layer, and the ability to learn through experience give the multilayer perceptron its computing power. **For these reasons, a fully connected, multilayer perceptron network using a back-propagation training method is used in this research.** Section 2.1.6.1 discusses the back-propagation training method in more detail.

### 2.1.5 Learning.

“Learning is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded,” [Hayk94]. Learning simply means that the nodes are able to change their input-output behaviors in response to changes in the environment. There are several methods of learning. Each method can be supervised, unsupervised, or use a reinforcement stratagem.

### 2.1.5.1 Methods of Learning.

There are four basic methods of neural network learning: error-correction, Hebbian, competitive, and Boltzmann. Hebbian and competitive learning are inspired by biological systems. Error-correction learning is based on optimum filtering. Boltzmann learning is based on ideas from thermodynamics and information theory.

Error-correction learning attempts to minimize a cost function based on an error signal so that each neuron's actual response approaches the desired response in a statistical sense. Error-correction learning then becomes an optimization problem. A commonly used cost function is the mean square error (MSE) in which MSE is minimized. The goal is to stop training when the MSE is minimized. However, the MSE has many local minima before reaching a global minimum. If the system is stable, the mean square error stabilizes. The rate of learning affects stability – too high and the system is unstable, too small and the system takes an excessively long time to learn. The back-propagation algorithm is based on error-correction learning. **The error back-propagation method is used for this research due to its popularity and the fact that “it provides a *computationally efficient* method for the training of multilayer perceptrons,” [Hayk94]**

Hebbian learning is based on associative learning. “Positively correlated activity produces synaptic strengthening and uncorrelated or negatively correlated activity produces synaptic weakening,” [Hayk94]. If a neuron repeatedly attempts to cause another neuron to fire, then its efficiency to fire that neuron increases. Uncorrelated firings of neurons weaken their synaptic coupling.

Competitive learning is a system in which the neurons compete to be the one that fires. The winner is referred to as a winner-takes-all neuron since only a single output neuron is active at one time.

Boltzmann learning uses a system in which all neurons have binary values for their states (energies). All synaptic connections are symmetric and units are updated one at a time. No feedback exists. Units are picked and states changed with some probability and eventually the system settles to a certain low energy state. Mean-field theory (MFT) learning is a subset of Boltzmann learning that was developed because the Boltzmann machine suffers from slow learning. MFT learning employs a deterministic instead of a probabilistic neuron firing model.

#### **2.1.5.2 Classes of Learning.**

Supervised learning requires an external teacher that knows the desired output for each input. The network's parameters are adjusted according to the difference between the actual response and the desired response with the goal of fully emulating the teacher to correctly respond to all inputs. There is no external teacher to oversee the learning process for unsupervised learning. An ANN using unsupervised learning is often called self-organizing. An unsupervised network attempts to develop the ability to form internal representations for encoding features of the input and thereby create new classes automatically.

In non-learning and reinforcement learning, the NN receives a reinforcement signal to maximize a scalar performance index. This signal makes the desired action more likely to happen again. “The basic idea behind reinforcement learning is to learn

the evaluation function so as to predict the cumulative discounted reinforcement to be received in the future,” [Hayk94]. A reinforcement system seeks only to improve performance. This is done through trial and error, exploration, and delayed rewards. The primary difference between a supervised learning system and a reinforcement system is that the supervised system is evaluated and directed in how to change its behavior while a reinforcement system simply evaluates its performance without using external guidance. A reinforcement system must use trial and error to probe its environment and figure out how to change its behavior. Reinforcement learning systems are slowed down by the conflict between the exploration process and the desire to improve their behavior. Supervised systems do not suffer from this problem.

**Supervised learning is used in this research due to the unlimited amount of possible input classes.** A reinforcement system is unable to probe its environment, due to the nearly infinite number of possible inputs.

#### **2.1.5.3 Learning Tasks.**

The learning method employed is largely determined by the learning task that the neural network is to perform. The learning task classes are listed below [Hayk94]:

1. *Approximation.* The ANN approximates an unknown nonlinear input-output mapping,  $d = g(x)$ , where  $x$  is the input and  $d$  is the desired output. Supervised learning may be viewed as an approximation problem.
2. *Association.* There are two types of association tasks, autoassociation and heteroassociation. An ANN performing an autoassociation task stores a set of inputs and then, when presented with a noisy version, the ANN retrieves the

correct pattern. Heteroassociation tasks pair the set of inputs with a set of outputs. Autoassociation tasks use unsupervised learning while heteroassociation tasks use supervised learning.

3. *Pattern classification.* Inputs are placed into a fixed number of categories. The ANN is repeatedly trained with a set of inputs and the correct classification. New input patterns are then presented for the NN to classify. As described, this task uses supervised learning. When used in an unsupervised learning manner, the ANN must perform an adaptive feature extraction or cluster the information prior to performing classification. **This research is a supervised pattern classification task.**
4. *Prediction.* These tasks require the ANN to predict a sample based on past samples. The goal is to model the underlying process as closely as possible. Often, error-correction learning is used to change free parameters to create a better model.
5. *Control.* The ANN controls another process. Difficult processes involving problems such as nonlinearity, noise, and many parallel actuators can be controlled.
6. *Beamforming.* Beamforming tasks are a type of spatial filtering used to locate a target signal through additive noise. The task of locating a single voice in a crowded room is a good example of this type of task.



### **2.1.6 Training of Neural Networks.**

Training of neural networks is a nontrivial problem. The “back-propagation algorithm represents a “landmark” in neural networks in that it provides a computationally efficient method for the training of multilayer perceptrons,” [Hayk94].

Learning requires many presentations of examples, or patterns, in a training set. A complete training set, or epoch, is a sequence of examples drawn independently at random until every pattern in that set has been used. The learning process continues until the error converges to some pre-determined minimum value. Randomizing the order of presentation of training examples helps to “roll past” local minima, or saddle points. Furthermore, the training examples should contain both positive and negative examples.

#### **2.1.6.1 Back-propagation.**

Back-propagation is an error-correction learning technique that offers a computationally efficient method for training multilayer perceptron networks. The goal of back-propagation learning is the generalization of an input-output relationship. Back-propagation works by making two passes through the network: a forward and a backward pass. The forward pass works by keeping the synaptic weights fixed, applying a stimulus, and then measuring the actual response. The output is then compared to the desired response and an error signal generated. The error signal is then propagated backwards through the network and the synaptic weights and biases adjusted to achieve a more ideal response. Back-propagation is a gradient technique and not an optimization technique. It performs a stochastic gradient descent in weight space for pattern-by-

pattern updating of synaptic weights and is simple to compute locally. Computations of each neuron are influenced only by those neurons that are in physical contact with it.

Another benefit of back-error-correction learning is that it permits graceful degradation, such as hardware failures, if small numbers of transient faults are injected into the network at each step. In other words, if the network is trained by adding faults, then the system becomes somewhat immune to small hardware failures and noise.

However, back-propagation is not without faults. For large networks, “the rate of convergence in back-propagation learning tends to be relatively slow, which in turn makes it computationally expensive,” [Hayk94]. This is because it uses the minimum amount of available information resulting in a slow convergence speed. With large networks, the training time can be prohibitively large and make the algorithm impractical. Experiments show that learning time scales exponentially with the number of inputs [Tesa88].

Local minima valleys can be problematic since back-propagation is a hill-climbing technique. The algorithm can get “stuck” in a valley and not progress towards the global minimum. Randomized the order of presentation from one epoch to the next somewhat reduces this problem and also tends to speed convergence of the network. Randomly “jogging” the weights by a small amount after each training epoch can also help the network get past local minima. **For this reason, the weights are randomly jogged within 97 – 102% of their original values after each epoch.** An even distribution is not used so that the cumulative changes will differ from the pre-jogged values.

The back-propagation algorithm has another limitation in that it can only learn a static input-output mapping. However, this is well suited for pattern-recognition applications where both the input and the outputs represent spatial patterns. **This is another reason for choosing back-propagation as the training method.**

A problem suffered by learning algorithms is determining stopping criteria. A common convergence criterion is considering the network trained when the absolute rate of change in the MSE is about 0.1 to 1%. In more demanding applications, error rates as low as 0.01% may be used [Hayk94].

To outline the back-propagation method:

1. *Initialization.* Start with reasonable network configuration. Set all synaptic weights and threshold levels to uniformly distributed, small random numbers.
2. *Present training examples.* Present an epoch. Steps 3-5 are performed for each sample in the epoch.
3. *Forward computation.* Compute the error signal where error is the desired response minus the real response.
4. *Backward computation.* Adjust the synaptic weights and biases by presenting the error signal.
5. *Iterate.* Iterate until the free parameters stabilize and the mean square error is acceptably small. Momentum and learning-rate parameters are typically adjusted (usually decreased) as the number of training iterations increase.

The back-propagation method used in this research relies on a sigmoidal activation function of the neurons which adds to the computational complexity of the

back-propagation algorithm. However, “the presence of nonlinearities is important because, otherwise, the input-output relation of the network could be reduced to that of a single-layer perceptron,” [Hayk94] which would severely limit its computational power. Analyzing the back-propagation method reveals that the computational complexity required to process each pattern is approximately twice that of the feedforward operation. However, this must be repeated on every pattern until the network is suitably trained.

Training examples are presented to the network using either pattern or batch modes. Pattern mode works by updating the synaptic weights and threshold levels after every example in the epoch. Batch mode updates the weights only after all of the examples in an entire epoch have been presented. Pattern mode requires less local storage and makes it less likely for the back-propagation algorithm to get trapped in a local minimum. However, batch mode training provides a more accurate estimate of the gradient vector. Simon Haykin offers some hints for increasing the performance of the back-propagation algorithm [Hayk94]. The tips that apply, to the network configuration used for this research, are outlined as follows.

1. An asymmetric, hyperbolic activation function may increase the learning rate.
2. Desired response values should fall within the range of the sigmoidal activation function. If this is not done, “the back-propagation algorithm tends to drive the free parameters of the network to infinity, and thereby slow down the learning process by orders of magnitude,” [Hayk94].
3. The synaptic weights should be initialized so that they are uniformly distributed inside a small range. This range must be carefully chosen. If the range is too

large the network saturates and produces small error gradients. If the range is too small, learning is initially very slow.

4. All neurons should learn at the same rate. This requires tweaking for each layer as the last layers typically have larger error gradients and learn faster.
5. Pattern mode training tends to be orders of magnitude faster than batch mode training, but is harder to parallelize.
6. As already mentioned, order of training examples in each epoch should be random.

Step 2 implies that **some type of normalization scheme should be applied to the inputs of a neural network to avoid over-saturating the neuron activation values. This is the basis for the use of a data transformation in this research.** Due to the reasoning in step 5, **pattern mode learning is used in this research since it tends to be much faster than batch mode learning. Constant neuron learning rate as described in step 4 is not used** since this can be difficult to determine, especially when multiple hidden layers are involved.

#### **2.1.6.2 Cross-Validation and Overtraining.**

There are an unlimited number of network models. Cross-validation validates the model and tests the performance of different network models. Cross-validation is not used to change parameters, but to aid in choosing a network model. Typically the data set is randomly divided into a cross-validation (testing) set and a training set. About 10-20% of the patterns are used to form a testing set that is used strictly for validation. The testing set is used to

“assess the performance of various candidate model structures, and thereby choose the ‘best’ one. The particular model with the best-performing parameter values is then trained on the full training set, and the generalization performance of the resulting network is measured on the testing set” [Hayk94].

Overtraining occurs when the network learns too many input-output mappings and memorizes the training examples. This causes loss of generalization, the ability to produce the correct output even for an input never before seen. This is similar to memorization in that the smooth input-output mapping is lost and replaced with a look-up table. An over trained network shows poorer performance on the cross-validation set and the MSE rises after reaching a minimum value as seen in Figure 2.3. The number of examples used for training should exceed the total number of synaptic weights divided by the fraction of errors permitted.

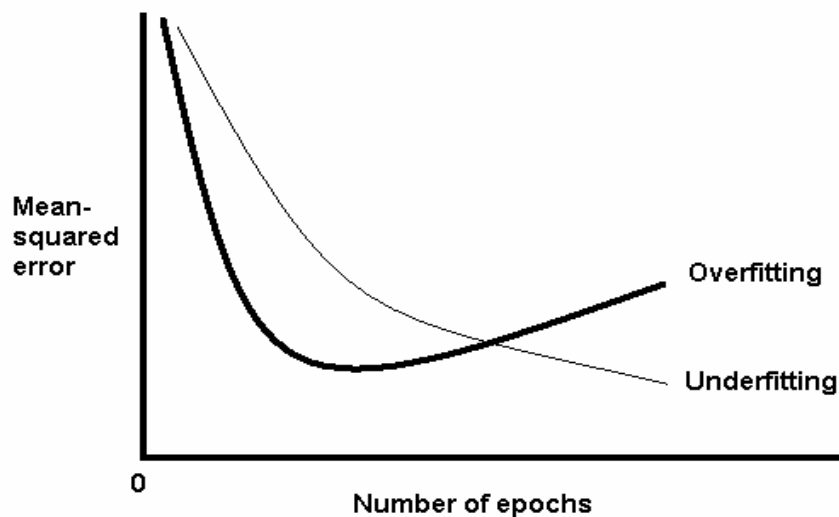


Figure 2.3. Examples of mean square error for an undertrained and overtrained neural network.

Cross-validation may also be used to adjust the learning rate. Once performance fails to increase by a certain amount, then the learning factor is decreased, typically by a factor of two. This process repeats until there is no further improvement and training is then stopped. **However, this research uses a constant learning rate.**

### 2.1.7 Network Models.

There are several common neural models. These models generally have a consistent network architecture, learning algorithm, and training method. This section identifies some of the more popular network architectures and briefly expands on each.

- Multilayered Perceptrons (MLP)
- Radial-Basis Function Networks (RBF)
- Recurrent Networks (includes Hopfield, Boltzmann, MFT).
- Self-Organizing (Kohonen)
- Modular Networks

“Multilayer perceptron networks are probably the most popular neural network architecture,” [Lisb92]. This is largely due to the back-error propagation algorithm used for training. “MLPs construct global approximations to nonlinear input-output mapping. Consequently, they are capable of generalization in regions of the input space where little or no training data are available,” [Hayk94]. Multilayer perceptron networks have been used successfully in solving optical character recognition problems.

Radial-basis function networks typically only have a single hidden layer and only construct local approximations to nonlinear input-output sets. This gives RBF networks

fast learning and less dependence on the order that the data is presented at the expense of memory requirements. RBF networks also suffer the dimensionality problem and the number of hidden nodes increases exponentially with the size of the input space. This is especially problematic in trying to solve large problems such as image and speech recognition. Additionally, RBF networks require some type of prior information built into the design of the network. **For these reasons, RBF networks are not used in this research.**

Recurrent networks are characterized by abundant feedback, symmetric synaptic connections, and nonlinear computing units. The Hopfield network acts as a nonlinear associative memory and retrieves patterns based on an incomplete or noisy version. The Hopfield network uses deterministic firing mechanisms, no hidden layers, and operates in an unsupervised manner. It may suffer from spurious states and unstable memories. The Boltzmann network is another recurrent network, but uses binary state neurons and a stochastic firing mechanism and can operate in a supervised manner. It also permits the use of hidden neurons which may serve as feature detectors. Another useful feature is that it avoids the “saddle point” problem through the use of a relaxation technique. This technique allows the machine to continue moving downhill until a certain number of attempts to minimize its energy state fail. However, the Boltzmann machine is computationally intensive.

Another type of recurrent network is the mean-field theory (MFT) machine. It attempts to improve upon the Boltzmann machine by using analog neurons that use a deterministic firing mechanism. In tests, the MFT machine has more memory storage and performs better at error-correction than any other network [Alsp92]. However, since



the MFT network uses only a single hidden layer, the number of applications that it is suitable for is somewhat limited [Hayk94]. Also, the MFT and multilayer perceptron networks have approximately the same performance when restricted to a single hidden layer and the output layer is much smaller than the input layer [Hayk94]. **Since the MFT network seems to offer no advantages over the MLP network, and because the single hidden layer may restrict its usefulness, the MFT network is not used for this research.**

Self organizing Kohonen networks are especially suited to modeling time-dependant processes. Kohonen networks use lateral connections between neurons in the same layer which are similar to connections in the cerebral cortex. Kohonen networks create self-organizing feature maps that fall into the class of vector coding algorithms that optimally place a fixed number of codewords into higher-dimensional space. This is useful for data compression. **As ideal as the self-organizing network seems, it does not fit the research task and is not used.**

The final type of network model considered is the modular network. Modular networks attempt to break computation down into pieces under the premise that it is easier for multiple specialized modules to specialize in certain tasks. It is easier for modular networks to learn multiple simple functions rather than an entire complex function. However, modularity is an additional variable and the scope of knowledge in this area is limited. **For this reason, a modular network is not used for this research.**

Regardless of the model used, there is always the risk of creating an unnecessarily large network. Starting with a small multilayer network and only adding new layers when the network cannot meet the input-output requirements is a possible solution.

Another approach is using a large network and then weakening or removing chosen synaptic weights. However, pruning a neural network can be very computationally expensive as cross-validation must be used to verify that the network's performance is still satisfactory after each pruning operation in addition to the added overhead imposed by the pruning function.

### **2.1.8 Pattern Recognition.**

Optical character recognition (OCR) is a simple vision problem, yet is difficult for machines to accomplish. A multilayer perceptron network has been used to solve this problem [LeCun90a]. The input layer consisted of 400 nodes – one for each pixel of the 20x20 pixel area processed. The system used four hidden layers and an output layer of ten nodes. The ten output nodes correspond to the numbers, 0-9, recognized by the neural network. In order to limit the number of free parameters, only the output layer was fully connected with the previous layer and weight sharing among synapses was used. Weight sharing allows several synapses to be controlled by a single weight. After some tweaking of the network, reject rates as low as 12-13% were achieved when allowing a 1% error rate.

Other configurations of neural networks have also tackled the OCR problem. Another system used a multilayer back-propagation network with four hidden layers. The final layer was fully connected and used shared weights. However, the hidden layers were arranged to use feature maps. The final network had a total of 4,635 neurons, 98,442 connections, and 2,578 parameters. After training on 30 epochs of nearly 10,000 examples, a reject rate 1.1% was achieved while allowing only a 1% error rate. When

tested with only the handwritten set, the system had a 9% reject rate. Some of the images were ambiguous even to humans [LeCu89].

### **2.1.9 Hardware Implementation.**

Very-large-scale-integration (VLSI) hardware can realize neural networks. VLSI hardware is massively parallel which works very well for neural networks. Even as far back as 1990, a VLSI-based system called CNAPS (Connected Network of Adaptive Processors), was capable of 1.6 billion multiple accumulates per second (GMACs) for a 25 Mhz chip. These chips could also be used in parallel. Training that would take about 4 hours on a 40 MHz SUN SPARC workstation was reduced to only about 7 seconds. The SUN workstation employs a RISC processor that can, at best, perform only a single instruction per clock cycle while a parallel VLSI implementation does not have this implementation.

Current field programmable gate arrays (FPGAs) are much more powerful. The latest Altera FPGA, Stratix II, is advertised to run at 370MHz and perform 284 GMACs. These gate arrays can also be used in parallel.

## **2.2 Wavelet Transform**

This section gives a brief review of the wavelet transform. It is often advantageous to work in the frequency domain for signal processing. The Fourier transform (FT) is a popular technique to transform a signal from the time domain to the frequency domain. The frequency component of the Fourier transform relates to the rate of change. For imagery, the frequency component corresponds to the changes in intensity variations. Low frequencies correspond to smooth variations throughout the image while high frequencies correspond to noise and sharp changes such as the edges of objects.

### **2.2.1 Fourier Transform.**

The Fourier transform reveals how much of each frequency component exists in a signal, but does not tell when in time those components exist. Heisenberg's uncertainty principle states that a particle's momentum and position can not be known simultaneously. This principle also applies to signal analysis. That is, perfect time and frequency resolution cannot be known for the entire signal. The Fourier transform is suitable for stationary (non-changing frequency) signals, but less useful for non-stationary signals. Almost all biological signals, such as ElectroCardioGraphies (ECG), ElectroMyoGraphies (EMG), and ElectroEncephaloGrams (EEO), are non-stationary.

The Fourier transform decomposes a signal into complex exponential functions of different frequencies according to the functions in Equations 2.4 – 2.6.

The Fourier transform of  $x(t)$ :

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-2j\pi ft} dt \quad (2.4)$$

The Inverse Fourier transform of  $X(f)$ :

$$x(t) = \int_{-\infty}^{\infty} X(f) \cdot e^{2j\pi ft} df \quad (2.5)$$

The complex exponential term can be written as a sum of sines and cosines as follows:

$$e^{2j\pi ft} = \cos(2\pi ft) + j \sin(2\pi ft) \quad (2.6)$$

The time at which the frequency component appears is not important because the integral is evaluated over all time (negative infinity to infinity). Thus, regardless of where the frequency component occurs in time, the result is the same. This is the fundamental reason why the Fourier transform is unsuitable for non-stationary signals.

Other transforms can provide both time and frequency information. The short-time Fourier transform (STFT) overcomes this shortcoming. The STFT works by using fixed sized windows multiplied by the FT using the equation:

$$STFT_X^{(\omega)}(t, f) = \int [x(t) \cdot \omega^*(t - t')] \cdot e^{-j2\pi ft} dt \quad (2.7)$$

where  $\omega(t)$  is the window function and  $*$  is the complex conjugate.

The problem with the STFT is that the windows are a fixed size for the entire signal and therefore usually either under or over-resolves frequency information at the expense of time information.

### 2.2.2 Wavelet Transform.

The wavelet transform is a relatively new transform tool, gaining recognition in 1987 as a new approach to multiresolution theory. “Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including subband coding from signal processing, quadrature mirror filtering from digital speech recognition, and pyramidal image processing,” [Gonz02]. The wavelet transform overcomes the time-frequency resolution problem.

The wavelet transform resembles the FT in that the signal is multiplied with some function. However, instead of using a complex sinusoid composed of sines and cosines for the frequency parameter, the wavelet transform uses a small wave (wavelet) which is not limited to just the sine and cosine functions. These wavelet, or basis, functions are defined over a finite interval and have an average value of zero. The basic idea of the wavelet transform is to represent any arbitrary function,  $f(t)$ , as a superposition of a set of such basis functions. These basis functions are obtained from a single prototype wavelet called the mother wavelet, by dilations or contractions (scaling) and translations (shifts). The wavelet coefficient value is a measure of the correlation between the wavelet and the signal at the current scale.

The primary difference between the STFT and the wavelet transform is that the wavelet transform changes the width of the window for different spectral components. The wavelet transform’s major importance is its simultaneous representation of both time and frequency information. The time-domain signal is passed through several high-pass and low-pass filters to remove those elements of the signal. Each time this process repeats, some frequencies are removed from the signal. Though not perfect, the wavelet

transform provides better resolution in time and frequency than the short-time Fourier transform. A wavelet transform cannot determine the frequency component exactly at every instant of time, but can determine the frequency over an interval by trading frequency resolution for time resolution. High scale (low frequency) components of a signal can be resolved more accurately in frequency at the expense of time resolution while high frequencies (low scales) can be better pinpointed in time; however, the knowledge of the frequency becomes less precise. Most signals contain short duration high frequency components and long duration low frequency components for which the wavelet transform is well suited.

Figure 2.4 presents a visual representation of this idea. The area in each box is a constraint of the Heisenberg's inequality. Observe that wavelet analysis resolves low frequency better (short height box) at the expense of time resolution (wide box). The signal domain has good time resolution at the expense of frequency resolution while Fourier analysis is just the opposite.

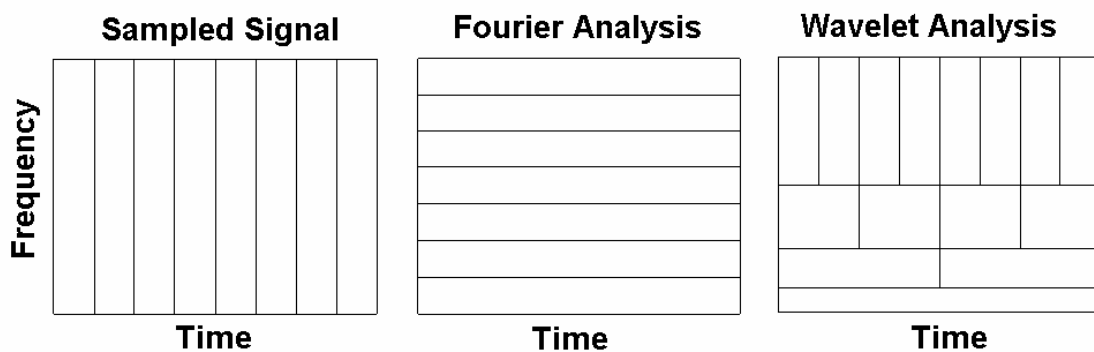


Figure 2.4. Time-frequency tilings of a sampled signal, Fourier analysis, and wavelet analysis.

### 2.2.2.1 Continuous Wavelet Transform.

The continuous wavelet transform (CWT) “transforms a continuous signal into a highly redundant function of two continuous variables – translation and scale.” [Gonz02] This redundancy is the result of overlapping that occurs when computing every possible scale and translation. The CWT is:

$$CWT_{\psi}(s, \tau) = \int_{-\infty}^{\infty} f(t) \psi_{s, \tau}(t) dt = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t) \psi_{s, \tau}\left(\frac{t - \tau}{s}\right) dt \quad (2.8)$$

where

$\psi\left(\frac{t - \tau}{s}\right)$  is the mother wavelet

$\tau$  is the translation (position)

$s$  is the scale

$\frac{1}{\sqrt{|s|}}$  is a normalization factor

Note that using  $s < 1$  compresses the signal while  $s > 1$  dilates the signal. The normalization constant is used so that each level has the same energy. The CWT is a reversible transform, meaning that an inverse transform recovers the original signal.

### 2.2.2.2 Discrete Wavelet Transform.

The discrete wavelet transform (DWT) “provides sufficient information both for analysis and synthesis of the original signal, with a significant reduction in the computation time,” [Poli04]. Instead of computing all scales and translations, the DWT uses principles of subband coding to analyze the signal at different scales. Typically



DWT uses dyadic (powers of two) upsampling and downsampling (subsampling) functions for each scale. Dyadic subsampling drops every other sample of the signal and halves the number of samples for each scale.

A two dimensional wavelet transform is necessary to take the wavelet transform of an image. A two-dimensional scaling function,  $\phi(x,y)$ , and three two-dimensional wavelets,  $\psi^H(x,y)$ ,  $\psi^V(x,y)$ , and  $\psi^D(x,y)$ , are required. Each of these is the product of a one-dimensional scaling function,  $\phi$ , and the corresponding wavelet,  $\psi$ . This results in a separable scaling function and three directionally sensitive wavelets:

$$\phi(x, y) = \phi(x)\phi(y) \quad \text{Separable scaling function} \quad (2.9)$$

$$\psi^V(x, y) = \psi(x)\phi(y) \quad \text{Vertically sensitive wavelet} \quad (2.10)$$

$$\psi^H(x, y) = \psi(x)\phi(y) \quad \text{Horizontally sensitive wavelet} \quad (2.11)$$

$$\psi^D(x, y) = \psi(x)\psi(y) \quad \text{Diagonally sensitive wavelet} \quad (2.12)$$

Note that one-dimensional products were excluded from the list of equations above.  $\psi^H$  is sensitive to horizontal edges,  $\psi^V$  is sensitive to vertical edges, and  $\psi^D$  is sensitive to diagonal edges. Equations 2.6 – 2.8 can be used to define a basic scaled function and a translated basic function:

$$\phi_{j,m,n}(x, y) = 2^{j/2} \phi(2^j x - m, 2^j y - n) \quad (2.13)$$

$$\psi_{i,m,n}(x, y) = 2^{j/2} \psi(2^j x - m, 2^j y - n) \quad (2.14)$$

where index  $i$  identifies the directionally sensitive wavelet,  $j$  defines the number of scales to compute, and  $j_0$  is the starting scale.

Finally, this allows the DWT of function  $f(x,y)$  with size  $M \times N$  to be written:

$$W_{\varphi}(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \varphi_{j_0, m, n}(x, y) \quad (2.15)$$

$$W_{\psi}^i(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \psi_{j, m, n}^i(x, y) \quad i = \{H, V, D\} \quad (2.16)$$

This produces an approximation of scale  $j_0$  of the signal  $f(x,y)$  (Equation 2.15)

Horizontal, vertical, and diagonal details for scales greater or equal to  $j_0$  can be calculated using Equation 2.16. Simply put, unique subbands are produced by filtering the image along the rows and then along the columns. These subbands are often referred to HH, HL, LH, and LL where the first letter denotes the filter function along the rows and the second letter denotes the filter function applied to the columns. For example, the LH subband executes a low pass filter along the rows and a high pass filter down the columns. This captures the horizontal edges. Similarly, HL captures the vertical edges, HH captures the diagonal edges, and LL provides a course, low-resolution approximation of the image signal. Figure 2.5 demonstrates a three scale (or level) wavelet transform of an image. Note that Equation 2.15 above gives the  $j_0 = 3$  scale approximation of the image which corresponds to the upper left block in the wavelet transform image.

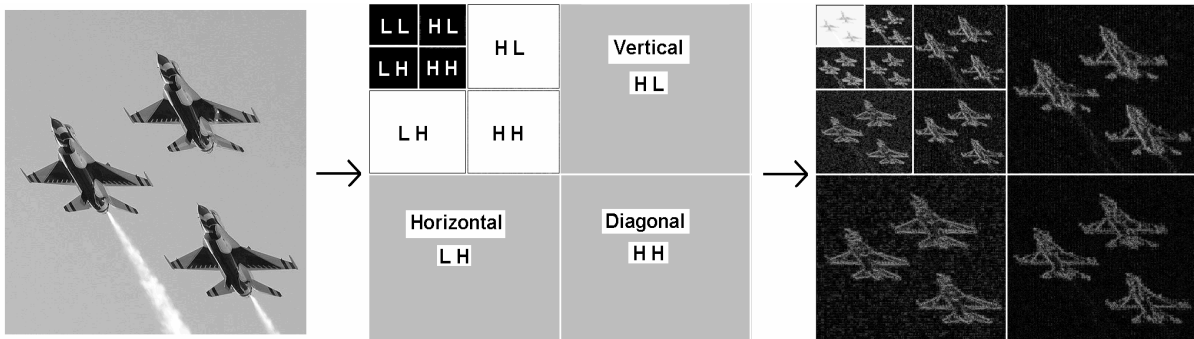


Figure 2.5. Three scale wavelet transform of an image.

The DWT transform is often used in image processing due to some of its desirable properties:

- *Locality*. The wavelet coefficients contain both time and frequency information.
- *Parsimony*. A few large wavelet coefficients contain most of the image information.
- *Clustering*. The wavelet coefficients tend to cluster together according to their magnitude.
- *Persistence*. Regardless of scale, the wavelet coefficients tend to be represented in the same manner. Basically, each scale displays the same image, but at a different resolution.
- *Multiresolution*. The signal can be analyzed at more than one resolution.

Figure 2.6 demonstrates parsimony and clustering as a histogram of the values of the wavelet coefficients. This graph preserves the sign of the wavelet coefficients for the log scaling to show how the coefficients cluster about zero. The left graph shows the raw wavelet coefficients while the right graph shows a sign preserved log scaling of the wavelet data using the equation,  $\log(1+|w|)$ . Note that most of the coefficient values are very small. Image compression uses this feature by discarding these small coefficients.

Generally, the largest valued coefficients appear at the LL subband of the highest scale (lowest resolution) as demonstrated in Figure 2.7. The wavelet coefficient values correspond to the closeness of match of the signal and the wavelet. The lower scales (higher resolution) fill in the details of the coarse approximation.

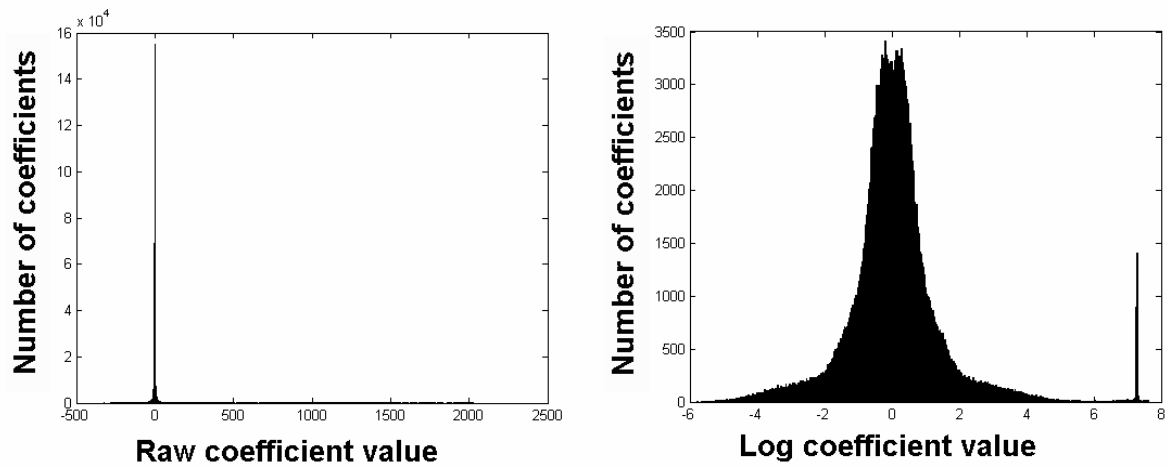


Figure 2.6. Histogram display of the distribution of wavelet coefficients.

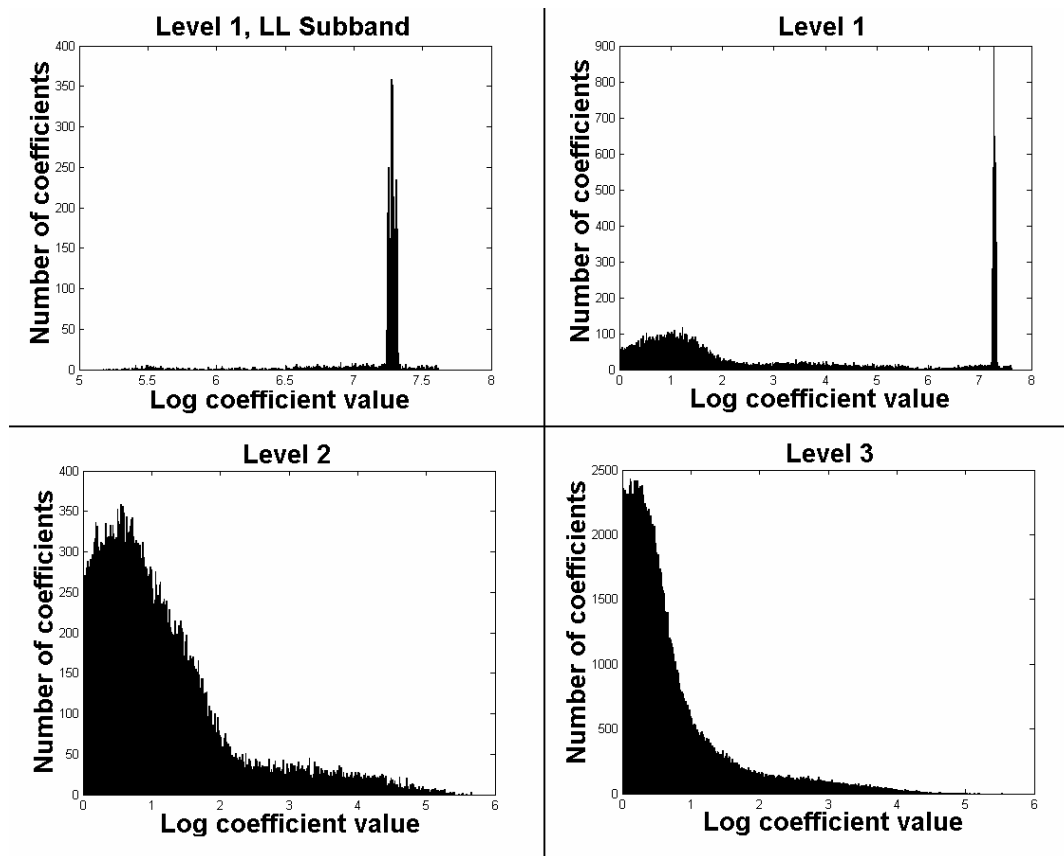


Figure 2.7. Histogram display of the distribution of wavelet coefficients for LL subband and across all three scales.

## 2.3 Object Recognition

“Many biological systems use vision as their most prominent way of gathering information about their environment” [Faug93]. Vision is such an important and difficult process that half of the entire cerebral cortex in primates is devoted to processing visual information [Fell91]. Computer vision is a subfield of artificial intelligence concerned with machine understanding of images. Often, computer vision uses explicit recognition in which an acquired image is compared to previously studied objects. Object recognition is a large part of computer vision, yet “recognition implies awareness of something already known,” [Kast91].

Vision poses a difficult task as digital images are simply a collection of numbers that measure light intensities on a surface. Brightness depends on an object’s shape, its material reflectance properties, and the light sources.

Vision problems are often grouped together because they must overcome some of the same challenges. However, object recognition is a larger problem than other vision tasks such as pattern recognition. This is due, in part, to the following reasons:

- View-dependent data must be matched with view-independent models.
- Lighting, material, surface geometries, and camera specific parameters must be overcome.
- Objects may be translated and/or rotated in three directions.

Section 2.3.1 discusses some of the difficulties of object recognition in more detail. Section 2.3.2 covers the general approach to object recognition.

### 2.3.1 Difficulties.

Computer vision is a relatively new field. Until about 10 years ago, commercial hardware capable of capturing and transferring full-resolution imagery at reasonable frame rates was not available. This problem was further compounded by lack of computational power to process the imagery. Although the mathematics were in place, the actual geometry of vision was not thoroughly understood.

Computer vision is plagued by a number of difficult problems. It is an inverse problem – ill-conditioned and difficult to solve without making some assumptions. The information required for a given task must be determined and then extracted from a two-dimensional image. This two dimensional projection loses an entire dimension from the actual object and must later be recreated in some form. The internal model representation of an object is not trivial. Multiple representations should be used to increase effectiveness or a model should be used that is invariant to three-dimensional rotation and translation.

Another problem is “how to discover all the information that is available from all the images and how to efficiently extract it through computation.” [Ma04] This feature extraction process is complicated by perspective, rotation, shape, distortion, and noise. Radial distortion is common. Radial distortion occurs when a short focal length is used, causing the straight lines in an image to curve. **Distortion and noise are limited as much as possible for this research.**

Lighting and reflectivity pose another problem since the machine only views an image as “a two-dimensional brightness array,” [Ma04]. Lambertian surfaces pose the fewest problems. A Lambertian surface diffuses light uniformly in all directions and

does not change appearance depending on the viewing angle. Examples of non-Lambertian surfaces are shiny surfaces and mirrors. **This research restricts models to Lambertian surfaces.**

Edge detection uses the principle of sharp changes in gradient to detect an edge. Edges come from different sources such as shadows, variations in reflectance, variations in texture, and variations in depth. However,

“the notion of a “peak” depends on the resolution of the image and the size of the window chosen. What appears as smooth shading on a small patch in a high-resolution image may appear as a sharp discontinuity on a large patch in a subsampled image,” [Ma04].

Basically, changing the resolution of an image affects the appearance of edges. Once edges are determined, the edges are often linked together to form a larger model. However, “edge linking is an extremely important, but difficult, step in image segmentation,” [Kast91]. An object that contains many regions that must be linked complicates edge linking.

Once geometric primitive features such as lines and points have been extracted, the correspondence between these geometric primitives must be determined. Most classical techniques can be divided into two categories: (1) local analysis such as contour following and (2) global analysis such as the Hough transform. Local analysis techniques fail in the presence of significant amounts of noise or incomplete data, but “global approaches are model-driven and will miss local features,” [Omid99]. Histograms provide a good example of a global technique that misses local features.

Humans solve the correspondence problem by viewing images using a combination of local and global analyses. Humans use surrounding cues, movement information (dynamics), neighboring structures, and naturally infer information about the scene even if part of the object of interest is occluded or outside of the visual field. However, machines are usually limited to only one type of analyses and loss of visual cues due to occlusion or exclusion from the image frame pose serious difficulty. Additionally, if a machine is only processing single, independent frames, then the machine has no chance to gather temporal information. **The objects in this research are not occluded and fit entirely within the frame of the image.**

### **2.3.2 Approach.**

Vision systems typically use the following sequence to convert raw pixel values into meaningful object information [Kast91]:

1. Image capture and enhancement
2. Segmentation
3. Feature extraction
4. Matching of features to models
5. Exploitation of constraints and image cues to recover information lost during the imaging process
6. Application of domain knowledge to recognize objects in the scene and their attributes

There are many geometric approaches, but all rely on several basic principles. One approach, using stereo images, is based on the epipolar constraint, in which two



image rays corresponding to the same three-dimensional point are coplanar. Another approach is the use of parallel lines as demonstrated in Figure 2.8. Yet another approach relies on perspective and vanishing points as pictured in Figure 2.9.

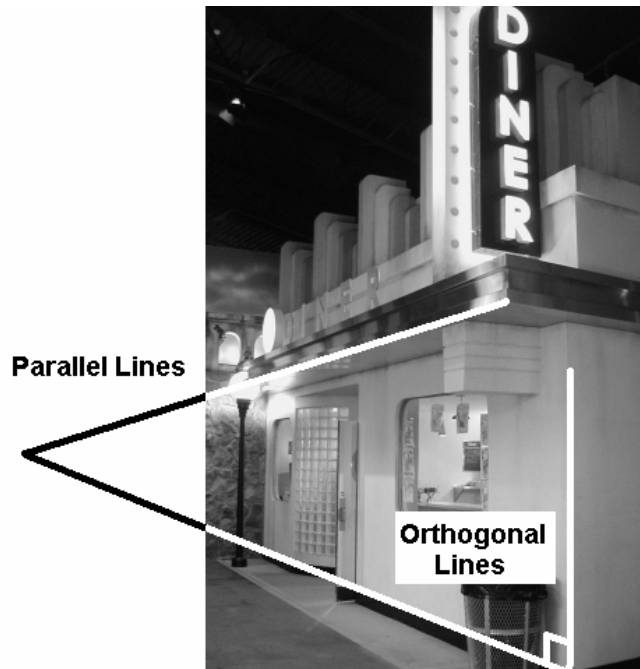


Figure 2.8. Parallel and orthogonal lines in an image.



Figure 2.9. Vanishing point in an image.

Low-level vision tasks include segmentation and boundary detection. These are necessary before conducting high-level vision tasks such as recognition and motion analysis.

Segmentation involves isolating objects of interest in an image and is often done either using edge detection or by methods that group pixels according to certain similarities. Similarities such as spatial proximity and intensity similarity can be used. However, domain knowledge must be applied since these are rarely consistent across an object. Intensity variations occur in a wide range of scales for natural images. Edge techniques are impaired by noise and intensity changes across an image.

The geometry is well understood for projecting geometric primitives, such as lines and points, onto an image plane. However, the inverse problem of extracting geometric primitives from an image is difficult because extracting these geometric primitives is usually done using local analysis. Local analysis only examines a small portion of the image at a time, rather than the large global picture.

Corner detection and optical flow (motion feature tracking) are two methods that rely on point features. Another point feature method uses epipolar geometry. Epipolar geometry uses multiple viewpoints of the same scene to project a point along with two camera optical centers to form a triangle. Using multiple points in the image allows the calculation of the camera poses and the three-dimensional position of the points.

Line features use techniques such as line fitting or connected component analysis. Often, the Hough or radon transform is used for these techniques. Basically, after edge detection is done, these transforms “connect-the-dots.” Line features are useful because man made environments usually have orthogonal and parallel lines.

After extracting primary features from an image, additional object recognition problems are faced. The first problem is dimensionality. Dimensionality refers to the idea that one or more dimensions dominate the geometry of an object. For example, a high tension wire is one-dimensional while a sheet of paper is two-dimensional. Another problem is that the object of interest may be occluded, or partly obscured. This means that even fewer object features can be extracted from the image. Finally, the rigidity of an object may affect its shape. Deformable objects must still be correctly identified regardless of their current shape. **The objects used in this research are all three-dimensional and rigid.**

Representation schemes keep only the main features of an object. Object recognition involves matching this representation to models of known objects. Some approaches for representing models are the use of bounding curves, feature matching, and region-based descriptors, such as textures. Feature matching uses distinguishing features such as corners, inflection points, and curvature discontinuities. There are many different schemes for representing three-dimensional objects. Some examples are wire-frame, constructive solid-geometry, surface boundary, and surface representations. **This research relies on the neural network to internalize the object representation from training examples.**

Invariance is an important property for eliminating instability (dependence on the coordinates of features). Invariants are algebraic or differential. Algebraic invariants include angles and distances and can be defined on more than one feature. This increases stability, but as the number of algebraic invariants increase, a larger number of features are required to define the invariant. Additionally, this increases the complexity of

matching to internal object models. Differential invariants include curves and surfaces. Differentials are more immune to occlusion problems, but this comes at the cost of complexity because they are computed from derivatives. As the complexity of a transformation increases, higher order derivatives are required which lead to instability.

Application of domain knowledge increases the effectiveness of the machine vision system. Domain knowledge is a large factor in the success of human vision, yet also accounts for why human vision can be easily misled. Figure 2.10 shows an example of applied domain knowledge. Experience leads most people to believe that only a single, continuous line passes through the box while in fact there are two offset, parallel line segments.

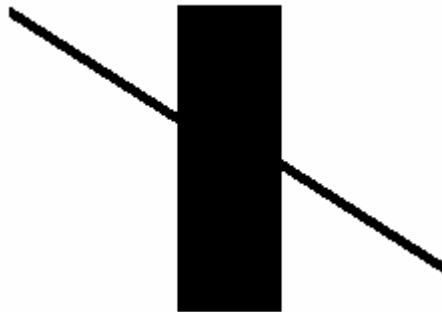


Figure 2.10. The line appears continuous due to applied domain knowledge.

## 2.4 Summary

The information presented in this chapter heavily influenced the methodology chosen for this research. The neural network model and formation of the object image database are both determined by information found in this chapter.

Neural networks (NNs) form an artificial intelligence system that is modeled after the nervous system. Neural networks can solve complex problems and problems that are difficult to formulate.

In Section 2.1 neural networks are discussed. Section 2.1 also highlights selections regarding the neural network model used for this research. The following list outlines these decisions. The section(s) discussing the decisions are provided.

- *General network architecture:* A fully-connected, feedforward, multi-layer perceptron network using two hidden layers. (Section 2.1.4)
- *Training method:* Supervised back-propagation (Section 2.1.4, 2.1.5.1 - 2.1.5.3, 2.1.6.1) with pattern mode learning (2.1.6.1)
- *Optimizations:* Weight jogging (97% - 102%) (Section 2.1.6.1) with a constant learning rate (2.1.6.2)
- *Network architectures considered and discarded:*
  - Recurrent networks (Section 2.1.4)
  - Radial Basis Function (RBF) networks (Section 2.1.7)
  - Mean Field Theorem (MFT) networks (Section 2.1.7)
  - Self-organizing network (Section 2.1.7)
  - Modular network (Section 2.1.7)

Section 2.2 discusses the basic properties of the wavelet transform, especially as applied to imagery. Also, the distribution of the values of the wavelet coefficients is discussed.

Object recognition is the topic of Section 2.3. The basic principles of machine vision are discussed as well as the fundamental difficulties. Finally, a generalized approach is discussed. Through Section 2.3 specific references regarding object recognition as used in this research are mentioned. These references are listed below along with the section in which they are located.

- Distortion and noise in images is limited as much as possible (Section 2.3.1)
- Only Lambertian surfaces are used to texture models (Section 2.3.1)
- Objects are not occluded and fit entirely within the image frame (Section 2.3.1)
- Objects are clearly three-dimensional (Section 2.3.2)
- Objects are rigid (Section 2.3.2)
- The neural network autonomously internalizes object representation models (Section 2.3.2)

### III. Methodology

#### 3.1 Approach

This chapter describes the methodology to meet the following goals:

1. Determine if the wavelet transformation is an efficient time and space pre-processor to a neural network for object recognition.
2. Determine which data scaling, if any, increases the success rate.
3. Determine which neural network topology proves the most successful.
4. Determine if the multiresolution aspect of the wavelet transform increases the success of the neural network.

To answer these questions, the following general process is followed:

1. *Creation of an image database.* An image database is created for eight objects in this research to provide training and testing sets for the neural networks. This is described further in Section 3.7.1.
2. *Wavelet transformation of each image.* A Daubechies (7,9) wavelet transform is performed on each image. Each subband forms a separate pattern set that is used for the corresponding subband's neural network. This is described further in Section 3.7.2.
3. *Scaling of the resultant wavelet transformed data.* After the wavelet transform, the resultant data is rescaled using several different methods before processing with the neural network. This is described further in Section 3.7.3.
4. *Determine optimal topology and data transform.* Cross-validation is used to determine the optimal data transform and neural network configuration.

Different neural network topologies are tested for each subband.

Additionally, each set of the rescaled coefficients from the previous step is tested for each of the neural networks. This is described in more detail in Section 3.7.4.

5. *Final training of the neural networks.* The variability charts from the previous step are examined and the best network topology and data transform selected. Further testing is conducted with several random seeds to obtain the “most average” trained network for each subband. This is described further in Section 3.7.5.
6. *Testing of the trained neural networks.* The CAD (computer aided design) testing set and the photo sets are tested with the trained networks. Additionally, the photo images most closely matching CAD model images are selected, cleaned, and tested. This is further described in Section 3.7.6.
7. *Correlate the outputs of the subband networks.* The three separate networks, one for each subband in a level, are used to form a larger neural network. The outputs of each of the subband networks serve as inputs into the larger network. This new network must also be trained. This is further described in Section 3.7.7.
8. *Testing of the final network.* The final, trained large network is tested. This is described further in Section 3.7.8.

Additionally, another test not included in this sequence is run to compare the system’s performance against a benchmark system that uses a single large neural network that processes the unmodified image patterns. Some simple tests are also run to examine the



performance of the selected network and scaling from step 4 above when using only photo pattern sets. These tests are further described in Section 3.7.9.

### 3.2 System boundaries and Overview

The system under test is an image correlation system that classifies an input image of an object for which it has been trained. It consists of several components: the wavelet transform, the scaling function, and the neural network. The feedback/training system for the neural network is part of the neural network component. Figure 3.1 shows a simplified model of the system.

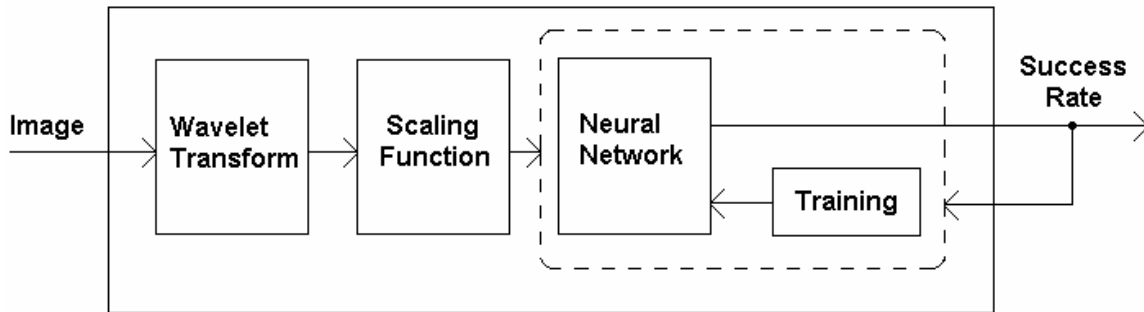


Figure 3.1. Simplified model of the image correlation system.

The Daubechies (7,9) wavelet transform is conducted on each of the images in the image database. Rescaling of the transformed image reduces the training time of the neural network as described in Section 2.1.6.1. The neural network uses the back-propagation training method which operates in a supervised learning mode. The neural network must be pre-trained on all of the objects for which it is required to classify. However, once the training of the network is complete, back-propagation is no longer required. The output layer of the neural networks consists of eight nodes, one for each

classification category. A selection among these eight nodes is made and compared to the desired output to decide if the object has been correctly classified. The ratio of correct decisions to the total patterns in the set forms the success rate for a pattern set.

### **3.3 System Services**

The system provides a single service. This service is a success rate. There are two possible outcomes: successful match and failed match. For this system, success is recorded when the system correctly identifies the object in the image. Failure occurs when the object is misidentified. Most detection systems also have false positive and false negative outcomes. However, for this research, the data set is constrained so that all images contain an object for which the neural network has been trained. This eliminates any false positive or false negative outcomes.

### **3.4 Performance Metrics**

There are two primary metrics for this system: (1) percent of objects correctly identified (success rate) and (2) mean square error rate. These metrics are chosen to compare the post-wavelet data transforms and network topologies and gauge the performance of the system. Mean square error is not used to determine the success of the system. It is an internal metric used only to help determine when the neural network is satisfactorily trained as explained in Section 2.1.5.2.

### **3.5 Parameters**

The parameters list all the possible variables that must be considered. Each parameter has some impact on the system's performance.

#### **3.5.1 System Parameters.**

Each component of the system has several parameters. The wavelet transform component has many possible parameters. The post-wavelet transform has a single parameter (i.e., which transform function to use, if any). The neural network contains the most parameters. The neural network is considered in two parts, the architectural aspect and the learning mechanism.

The wavelet component has two parameters, the wavelet function and the number of decompositions. The number of decompositions determines the number of subbands created and therefore the number of neural networks required. Additionally, the wavelet algorithm determines how those coefficients are calculated. These parameter variations affect the performance of the neural network in an unknown way since the effects are difficult to determine due to the complex nature of neural networks.

The neural network architecture has many parameters that affect its performance in undetermined ways. Some of the architectural parameters are (1) the type of neural network, (2) number of layers, (3) number of neurons in each layer, (4) connectivity, (5) activation (firing) function, (6) the neuron updating model, and (7) the winner selection scheme. These all affect performance. Section 2.1.4 explains that increasing the number of hidden layers may help the neural network to generalize at the expense of increasing

training time and instability. The winner selection scheme determines the way in which the network outputs are interpreted.

The learning mechanism for the neural network also has several parameters. Some of these learning mechanism parameters are the learning method, learning rate, and synaptic weights initializations. The learning rate must be set low enough to maintain stability while still permitting the neural network to learn at an acceptably fast rate. Synaptic initialization values can significantly affect the performance of the neural network.

Additionally, there are several parameters for presenting data to the neural network. The number of samples per epoch, number of epochs, order that the samples are processed in each epoch, stopping criteria, and random weight jogging are additional parameters. These parameters affect the training of the neural network which determines the performance of the neural network. Generally, the more samples used for learning, the better the neural network performs. However, overtraining can result in poor performance. Section 2.1.6.1 lists some criteria for network convergence. Weight jogging is a scheme in which the weight values are randomly “jogged” by a small amount after each training epoch, usually within several percent of their original values. This has been noted to help a neural network get past local minima [SNNS].

To summarize the system parameters:

- Wavelet:
  - Wavelet function
  - Levels of decomposition
- Scaling Function

- Neural network (architecture)
  - Type of neural network
  - Number of hidden layers
  - Number of neurons in each hidden layer
  - Connectivity
  - Activation (firing) function
  - Neuron updating model
  - Winner selection scheme
- Neural network (feedback/training)
  - Learning method
  - Learning rate
  - Synaptic weights initializations
  - Samples per epoch
  - Number of epochs
  - Order of presentation of samples
  - Stopping criteria
  - Weight jogging

### **3.5.2 Workload Parameters.**

The workload for this research is the individual images, not the way in which they are grouped or processed. The workload has the following possible parameters:

- Number of objects
- Image model (real-life model / CAD model)

- Size of images (resolution in pixels)
- Bit depth (color / grayscale)
- Noise (background / object)
- Rotation angle of model
- Elevation angle of model
- Scale of the model within the image

The workload parameters affect all aspects of the system. The number of objects and the number of images per object affect the overall size of the image database that must be constructed. However, there must be enough object examples in order to properly train the neural network.

Simple geometric blocks are used for the object models. Images of these models are captured from a CAD program and from photographs of real-life models. The CAD images represent the ideal case with minimal noise and no shadows that may hamper the performance of the neural network.

Image resolution must be sufficient so that important details such as edges and corners can be recognized and detected by the system. Higher resolution images result in larger matrices of wavelet coefficients. This may increase the performance of the neural network since it has better resolution to identify the interesting features (edges and corners), however, the added computational complexity of the neural network quickly limits the size. Higher bit-depth images may provide the neural network with more information to successfully match objects, however this comes at the cost of added computational complexity. Background noise and shadows in the real-life model images

may “distract” the neural network, decreasing performance or may actually provide additional information

The quality of the images, brightness, and contrast likely affect performance because sharper images exhibit more distinct variations in the wavelet coefficient matrix. The CAD models are surfaced with various textures to reduce the chance of the neural network keying on a particular surface feature.

The rotation angle and elevation angle determine how the object appears in the image. Both rotation and elevation angles have unlimited possibilities as the object may be rotated by any arbitrary amount. Distance from the object is also a factor as this affects the scale of the object’s appearance in the image.

### **3.6 Factors**

The factors are carefully selected from the parameters to limit the scope of the research.

#### **3.6.1 System Factors.**

The wavelet decomposition employs a three level Daubechies (7,9) tap wavelet. Three decomposition levels are commonly used in image processing and provide subband matrices small enough to be handled by the neural network.

The neural network parameters are selected to maximize performance of the neural network. As mentioned in Section 2.1.7, feedforward, multilayer perceptron networks with two hidden layers have been used successfully for other image processing applications. Additionally, two hidden layers provide reasonable training times and

stability. More hidden layers often require significantly longer training times before the network settles. Back-propagation provides an efficient training method for this type of network. Therefore, a fully connected, feedforward, multilayer perceptron neural network with two hidden layers is used for this research. Back-propagation is the selected method of training.

Most of the feedback and training parameters for the neural network use the default methods provided by the SNNS (Stuttgart Neural Network Simulator) which is the neural work tool used for this research. SNNS employs, by default, the most favorable settings for the selected network. The synaptic weights are initialized to small distributed random values between -1 and 1 as suggested in Section 2.1.6.1. Pilot studies show that a value of 0.50 allows the network to train reasonably fast without introducing excessive instability.

The winner-takes-all selection scheme determines the classification made by the neural network. Winner takes all is a simple method that is often used for classification problems. The output node with the highest output value is selected as the winner. However, a weakness of this method is that the network does not indicate if the output values are close in value.

The data set determines the number of samples per epoch and the stopping criteria determines the number of epochs processed. The point at which training is considered complete is a function of the testing set mean square error, the training set mean square error, and the success rate. These criteria were specifically developed for this research project. Section 3.7.6 discusses the details of the stopping criteria.



Randomly shuffling the patterns after each epoch eliminates the order of presentation of samples as a factor. Weight jogging was specified to be within -5% to +2% of the original values so that the cumulative effect of jogging will slightly change the overall weight values. Jogging is performed before each epoch is processed for training.

To summarize the factors chosen:

- Wavelet:
  - Wavelet function: Daubechies (7,9)
  - Levels of decomposition: 3
- Scaling function: none, linear, log, both
- Neural Network (Architecture)
  - Type of neural network: multilayer perceptron
  - Number of hidden layers: 1 or 2
  - Number of neurons in each layer
    - Input: determined by size of subband matrix
    - Hidden layer 1: 64, 128, 256
    - Hidden layer 2: 0, 16, 32, 64, 128 (if 0, then no hidden 2 layer)
    - Output: 8 (equal to the number of objects)
  - Connectivity: fully connected, feedforward
  - Activation (firing function): logistic

$$a_j(t) = \frac{1}{1 + e^{-(net_j(t) + \theta_j)}} \quad (3.1)$$

- Neuron updating model: topological order

- Winner selection scheme: winner takes all (WTA)
- Neural Network (feedback / training)
  - Learning method: back-propagation (supervised)
  - Learning rate: 0.5
  - Synaptic weight initializations: randomly distributed values between -1 and 1
  - Samples per epoch: determined by number of patterns in each pattern set
  - Number of epochs: determined by stopping criteria
  - Order of presentation of samples: randomly shuffled after each training epoch
  - Stopping criteria: based on training set MSE, testing set MSE, and success rate (see Appendix E)
  - Weight jogging: Performed before every training epoch, -5% to +2% of original values

### **3.6.2 Workload Factors.**

Workload factors selected present the system with a wide range of possible inputs and determine the system's performance under different conditions. All of the photo images are of good quality, have little noise, and have good contrast and brightness. The CAD images are artificially generated and are excellent quality images with no noise and superb contrast and brightness.

Eight objects are used to test the performance of the system. These objects were modeled from toy blocks due to their geometric simplicity and availability for real-life

photographs. The CAD models are surfaced with four different textures in an effort to prevent the neural network keying on any particular surface feature. The CAD modeled objects using a smooth surface are pictured in Figure 3.2. Images of the real-life models of the corresponding models are shown in Figure 3.3.

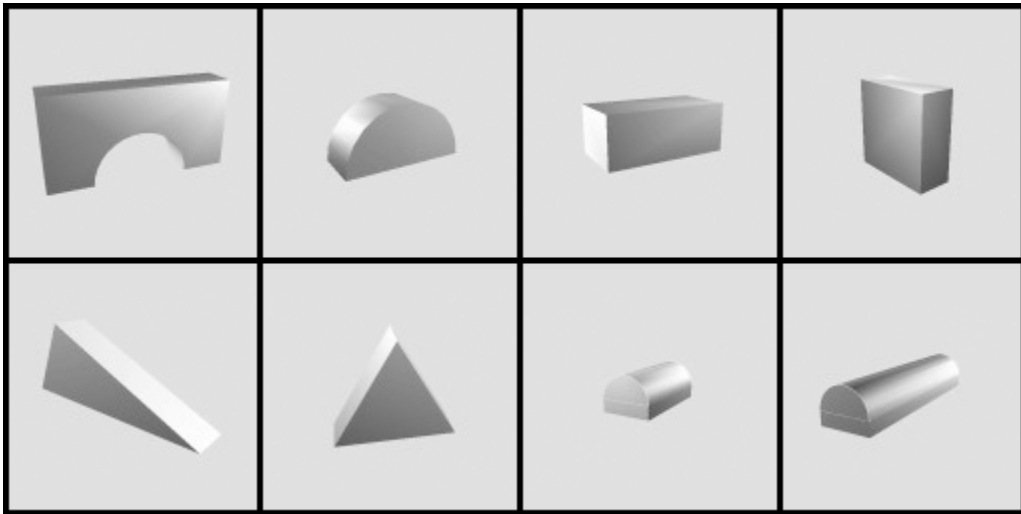


Figure 3.2. Smooth textured CAD models of the objects used in this research.

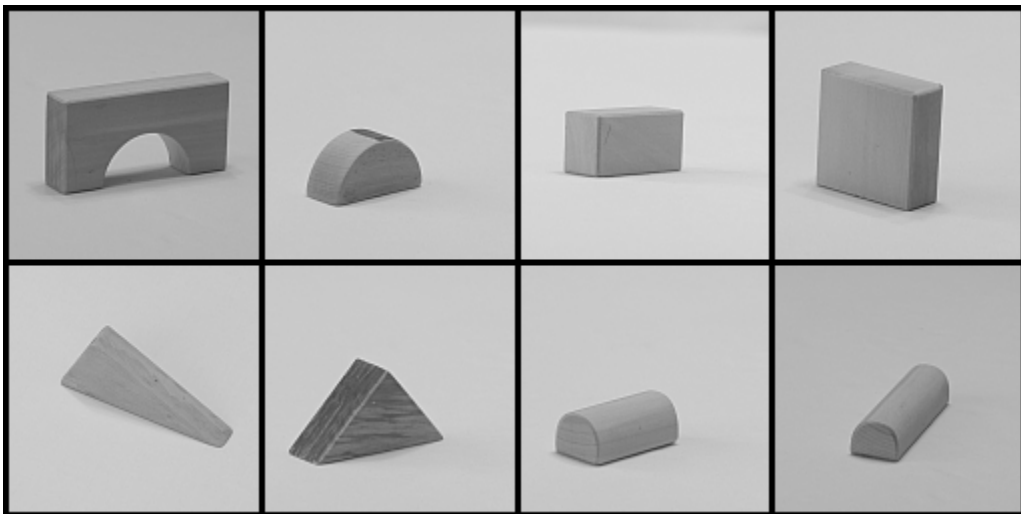


Figure 3.3. Photos of the real-life models.

The CAD model and real-life model of the triangle model differ slightly (bottom row, second from left) in order to test the generalization of the network. However, as the two models are the most similar of the eight, system performance should not be significantly impacted. If performance is impacted, it would only be for objects that should be classified as this triangle.

The implementation of the wavelet transform used in this research requires the image size be a power of two in both dimensions. Resolution of the images is limited to 128x128 pixels. Pilot studies using 256x256 images demonstrate that training time of the neural network was too long for this research effort. Using 128x128 pixel images and taking 3 wavelet decompositions yields pattern sizes as small as 256 inputs (16x16) and as large as 4096 inputs (64x64). This restriction of image size is applied to both CAD and photograph images. All images are converted to 8-bit grayscale to further reduce the number of neural network inputs. True color (24-bit) images would require three times as many inputs since each color plane would have to be represented.

Every attempt is made to photograph the real models as distinctly as possible without adding background noise. This is done by using a plain white background and bright overhead lighting. Shadows are a problem in the photographed model set and may impact system performance. The CAD models are surfaced with four different textures as shown in Figure 3.4 to reduce the chance of the neural network keying on a particular surface feature. One of the surface textures chosen was a wood-grain pattern very similar in texture and color to the photographed model.

In both the CAD and photo sets, the object are rotated about one axis and approximately 20-25 images are captured per full rotation. The elevation angle is also

varied, but remains above the base plane so that the bottom of the object is never visualized in any of the images. Four different elevation angles are used. Duplicate and difficult images are discarded. Difficult images are those in which a human has a difficult time properly classifying the object or images in which only a single face of the object is seen. The final result is 377 photographed images and 2151 CAD model images. Since training occurs only with the CAD images and testing with the photograph images, it is acceptable that the number of CAD images outnumber the photographed images.

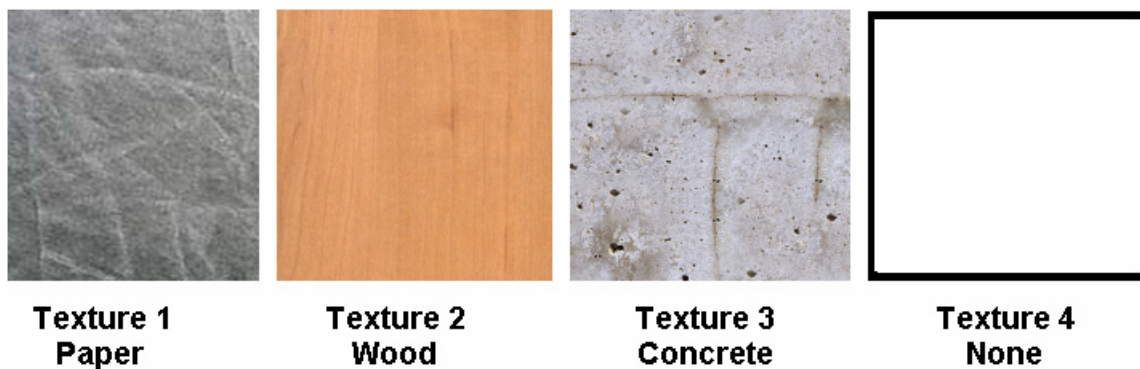


Figure 3.4. Textures used to surface CAD models.

The image sets are divided up into a testing set and a training set. Usually 10-20% of the total data set is used for cross-validation (the testing set) in order to evaluate the performance of the network model as described in Section 2.1.6.2. For this research, testing sets consist of approximately 10% of the total available patterns that are randomly selected.

Scale is a difficult factor to control for the photo set. Keeping the object centered as the object is rotated is problematic. The camera zoom must be adapted as each

photograph is taken so that the object fills the frame as entirely as possible without extending outside of the frame.

To summarize the workload factors:

- Number of objects: 8
- Image models: CAD and real-life models
- Resolution of images: 256x256 pixels
- Bit depth: 8-bit grayscale
- Noise in background and/or area surround target: All objects are photographed on a uniformly blank background. Four different textures are used on the CAD models.
- Rotation angle: For CAD models, approximately 20-22 images per elevation angle per texture for each object. For photographed models: 8 for each elevation angle for each object.
- Elevation angle: 4 per object
- Scale: object fits within image frame

### **3.7 Approach in Detail**

This section expands on the brief overview presented in Section 3.1.

#### **3.7.1 Creation of an Image Database.**

Images of objects used for this research are modeled from real-life objects. The object models are measured and a CAD model created for each object with the exception of the isosceles triangle as noted in Section 3.6.2. The models are textured with four

different textures (pictured in Figure 3.4) to prevent the neural network from keying on any particular surface feature. Light sources are added to the three-dimensional view and the object is rotated about a single axis. Approximately 20-25 images are captured as the model is rotated. After each full rotation of the object, the elevation angle is changed and the object rotated again. This process repeats for each of the four textures.

IrfanView, a freeware image processing program for the Microsoft Windows platform, is used to perform the image captures and other image manipulations [Irfan03]. Image captures were initially 520x520 pixels to fully capture the window area in the CAD program. After the image capture, the images are rescaled to 128x128 pixels using a Lanczos filter. Finally, the images are converted to 8-bit grayscale. Due to the necessity of capturing a square image so that the rescaling function would not distort the image, some of the surrounding window was also captured. This “noise” is removed as the pattern files are created and replaced with the background color.

Photographs of the model objects are taken to form a testing set. The object is placed on a large table with good overhead lighting and a white background. A tripod-mounted, remotely activated, digital camera is used to take all photographs. The tripod and remote prevent camera shake which causes blurry images. The camera is equipped with a LCD (liquid crystal display) that displays the current frame which ensures that the model fits within the frame of the image.

### 3.7.2 Wavelet Transformation of Each Image.

The Daubechies (7,9) tap wavelet is used for the discrete wavelet transform. The Daubechies (7,9) wavelet transform uses the nine low-pass, decomposition filter coefficients listed in Equation 3.2 and the seven high-pass, decomposition filter coefficients listed in Equation 3.3.

$$\vec{l} = \begin{bmatrix} 0.03782845550726 \\ -0.02384946501956 \\ -0.11062440441844 \\ 0.37740285561283 \\ 0.85269867900889 \\ 0.37740285561283 \\ -0.11062440441844 \\ -0.02384946501956 \\ 0.03782845550726 \end{bmatrix} \quad (3.2)$$

$$\vec{h} = \begin{bmatrix} 0.06453888262876 \\ -0.04068941760920 \\ -0.41809227322204 \\ 0.78848561640637 \\ -0.41809227322204 \\ -0.04068941760920 \\ 0.06453888262876 \end{bmatrix} \quad (3.3)$$

The Z-transform equations of  $\mathbf{l}$  and  $\mathbf{h}$  can be written as in Equation 3.4.

$$\begin{aligned} L(z) &= l(0) + l(1)z^{-1} + l(2)z^{-2} + \dots + l(8)z^{-8} \\ H(z) &= h(0) + h(1)z^{-1} + h(2)z^{-2} + \dots + h(8)z^{-6} \end{aligned} \quad (3.4)$$

Convolving the rows and columns of the image with the appropriate decomposition filter coefficients,  $\mathbf{l}$  and  $\mathbf{h}$ , generates the wavelet coefficients for the first scale. For example,



to create the LL subband of coefficients, the rows of the image would be convolved with  $l$  and then the columns. LH would require the rows be convolved with  $l$  and the columns with  $h$ . Each scale is created by zero padding  $l$  and  $h$  (i.e., inserting a zero every other row).

The Daubechies (7,9) wavelet transform used in this research is based on a lifting scheme created by Wim Sweldens. This lifting scheme yields the same results, but is computationally more efficient. The code used in this research was written by Roger Claypool [Clay99]. An inverse wavelet transformation of the wavelet transform verifies that the original image is recovered.

Daubechies (7,9) is used because it is a biorthogonal wavelet, is pre-existing, and is widely accepted for image processing. Daubechies (7,9) is so widely used in image processing that it is part of the JPEG2000 standard. Details of the design and implementation of other wavelets is beyond the scope of this research.

### **3.7.3 Scaling Function.**

The wavelet transformed results are rescaled in the data scaling component. The scaling function operates independently on each subband of the wavelet transform.

The four possible scaling functions are as follows:

- *None.* No rescaling is performed and the wavelet coefficients are used “as is.” Each value is rounded to the nearest integer.

- *Linear.* The wavelet coefficients are normalized between -1 and 1 using the formula

$$f(x) = f(x) / (f(x_{MAX}) - f(x_{MIN})) \quad (3.5)$$

Each value is rounded to the ten-thousandths place.

- *Log.* The log transformation in Equation 3.6 is used, but sign is preserved.

$$f(x) = \log(1 + |f(x)|) \text{ for each element of } f(x) \quad (3.6)$$

Each value is rounded to the ten-thousandths place.

- *Both.* The log transform is performed and then linearly normalized. Both transforms are the same as those described above. Each value is rounded to the ten-thousandths place.

Rounding of the data values is necessary to ensure that the patterns are compatible with SNNS, to maintain smaller file sizes, and because nothing is gained by the extra precision. Typically, the raw wavelet coefficient values in each subband are on the order of  $\pm 10^3$ .

It is worth noting that the log transformation most closely follows the naturally exponentially distributed wavelet coefficient values. Linear normalization was used to scale the wavelet coefficients between -1 and 1 to avoid saturating the neuron activation values as explained in Section 2.1.6.1. However, due to the distribution of the wavelet coefficient values, linear normalization causes most of the values to be very close to zero, especially after values are rounded to the ten-thousandths place.

The results of these steps yields four patterns sets for each subband. Figure 3.5 displays the role of the data transform in the creation of the pattern sets for the neural network.

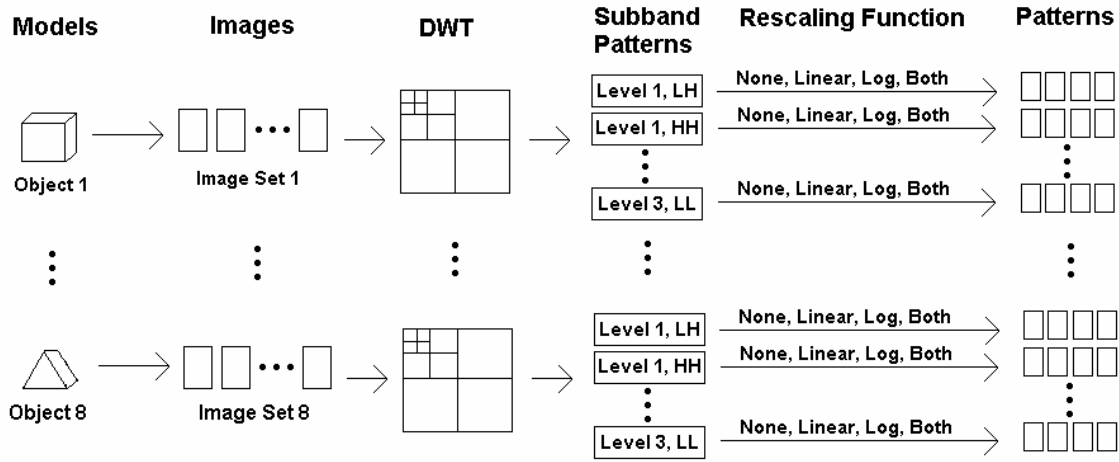


Figure 3.5. Creation of the pattern sets.

#### 3.7.4 Determine Optimal Neural Network Topology and Data Transform.

Neural networks for each subband are created using the `ff_bignet` tool that is included in the SNNS package. Fully connected, feedforward networks are created for each scale (256 inputs, 1024 inputs, and 4096 inputs). All networks have eight outputs, one for each object for which the network is trained. Pilot studies are conducted using 128 or 256 nodes in the first hidden layer and 0, 32, 64, or 128 nodes in the second hidden layer. Pilot studies conducted prior to this research revealed that the networks took an excessively long time to train when the first hidden layer contained 256 nodes. Also, the performance of the networks is significantly lower without a second hidden layer. Therefore, all tests are run with 128 nodes in the first layer, and 32, 64, or 128 nodes in the second hidden layer. Throughout this research, these networks are referred

to as 128-32, 128-64, and 128-128 where the first number is the number of nodes in the first hidden layer and the second number is the number of nodes in the second hidden layer.

After performing the wavelet transformation and data scaling on each image in a set, the set of patterns are used to train or test the network. An autonomous neural network which is independently trained and tested processes each subband in each level. Training is conducted by randomly shuffling the patterns in a set and processing the patterns one at a time until that epoch is finished. The patterns are then reshuffled and the process repeated until training is complete.

The neural network operates in a supervised manner and requires external feedback so that it can learn what constitutes success and failure. The back-propagation training method is the teaching method used to train the neural network. Each network for each subband is trained with each of the data transformed patterns at least four times using different random seeds for initialization and weight jogging. The networks are trained with the training set of patterns that consist of 90% of the total number of patterns and tested with the remaining 10% after every five training epochs. The networks are never permitted to train on the testing sets. These sets are used only for validation and gauging the network's performance.

Training is conducted for 120 training epochs which is more than sufficient to fully train the network as pilot studies show. Mean square error is recorded after processing every training or testing epoch. After conducting a test epoch, the classification is determined using a winner-take-all strategy on the outputs. This allows computation of the success rate.

### **3.7.5 Results of Trial Networks.**

A variability chart is created using the statistical software package, JMP v5. This allows for easy visual analysis to determine the best transform and network topology for each subband. Detailed statistical analysis is not conducted for this step as the goal is only to approximately determine the best performer.

The selected network topology is trained using the selected data scaling using a different random seed each time. An “average performer” is chosen as the final trained network. This is done to avoid using a network with an unusually favorable or poor performance. Also, the trained networks are more easily reproduced even if a different random seed is used.

As the average performing network runs, statistics including the success rate and the mean square error (MSE) for both the training set and the testing set are generated. A cubic spline model is fit to the MSE data using MATLAB. Starting at the first epoch, the training MSE and test MSE models are examined. Once the values for these two statistics drop and begin to level off, then the statistics for the success rate are examined. The network is considered fully trained once the average success rate for 10 samples is within 1.5% of the average of the remaining values through the 120<sup>th</sup> epoch. Figure 3.6 shows a basic representation of this process.

The network is only saved after every five training epochs due to the increased time and space that saving the network requires. Therefore, the next highest network save state is used for the final trained network for that subband (i.e., if the network is fully trained after 62 epochs, the network is saved after the 65<sup>th</sup> epoch is processed).

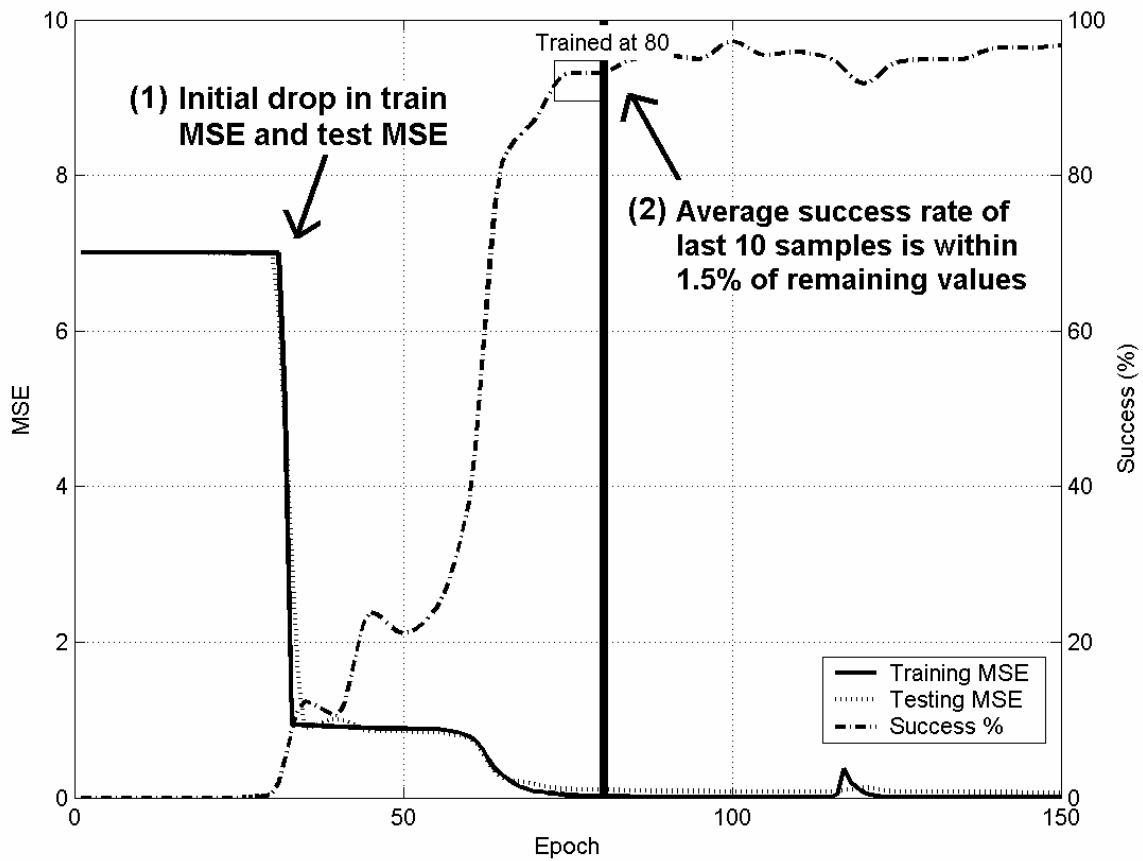


Figure 3.6. Determination of a neural network’s “trained epoch” (the epoch at which the network is fully trained).

### 3.7.6 Testing of the Trained Neural Networks.

Each subband network is tested with the CAD testing set and the set of photo images. Additionally, the CAD testing set is compared to the photo image set. Thirty-three (33) of the photo images that most closely match a CAD model are selected to form the good photo set. Rotation angle, elevation angle, and the scale of the object in the frame of the image determine these selections

### **3.7.7 Create a Network Composed of the Individual Subband Networks.**

The three separate subband networks in each level are used to form a larger “glue” neural network that correlates the outputs. The outputs of each of the subband networks serve as inputs into the larger network. Each of these new level networks is trained.

Three different architectures for the glue networks are created and tested. Figure 3.7 shows the correlating neural network. This network connects the corresponding node from each subband network. The other two architectures are fully-connected so that every output node from every subband network is connected to each node in the next layer. One of these two networks uses an eight node hidden layer between the subband network outputs and the final 8 node output layer. The other fully-connected network does not have a hidden layer, but connects the outputs of the subbands directly to the computational output layer.

Fifteen “glue” networks are tested in this manner. The three architectures described in the previous paragraph are used to connect the subbands in the following levels:

- Level 1
- Level 2
- Level 3
- Level 2 and 3
- Level 1, 2, and 3

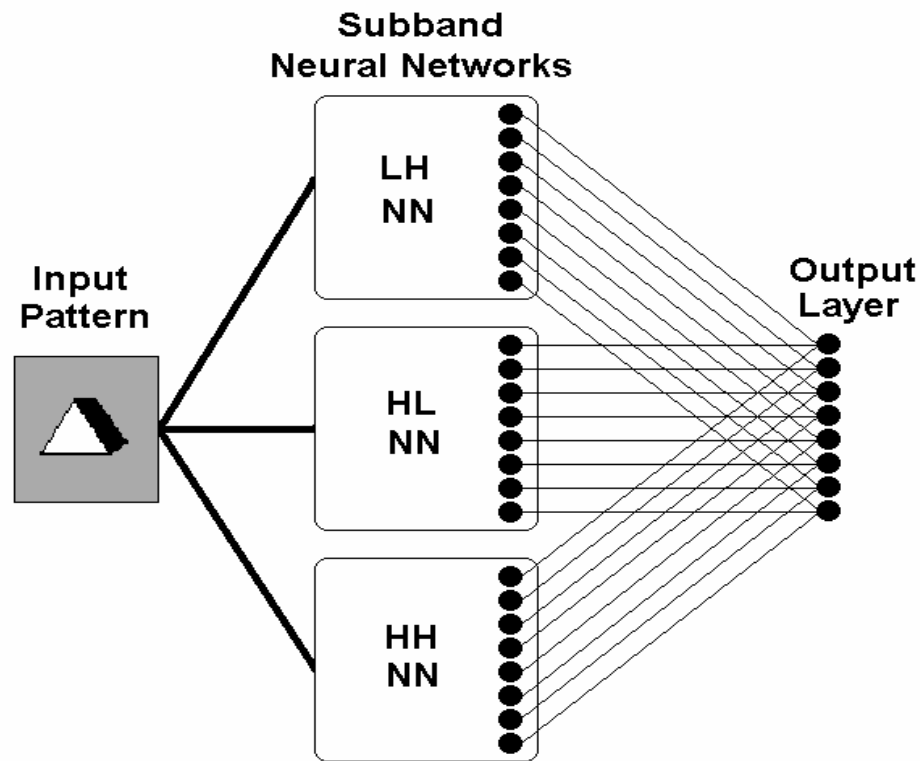


Figure 3.7. Large correlating network connecting the subband networks.

### 3.7.8 Testing of the Final Network.

The final trained networks consisting of the three smaller networks are tested and compared to the performance of the individual subband networks. The same testing sets used for the subband networks are also used for the large glue networks.

### 3.7.9 Additional Verification Tests.

Several additional tests are run to evaluate the performance of the selected network topologies and data transforms when trained and tested on only photo sets. The same 90-10 scheme (90% used for training, 10% for testing), is used to split the photos into two sets of patterns. Due to the amount of computational time, only the level 2 and



level 3 subband networks are trained and tested. However, 10 runs are performed with different random seeds to account for random variations.

Another test is performed to gauge the performance of a neural network on the non-transformed spatial images. A large fully-connected neural network is created and trained on the CAD training set. Testing is conducted using the CAD testing set. The network consists of  $512 \times 512 = 16,384$  inputs, 512 nodes in the first hidden layer, 128 nodes in the second hidden layer, and the usual 8 nodes in the output layer. Due to the huge computational requirements of this network, only a single run is performed.

### **3.8 Evaluation Technique**

The evaluation technique used is the measurement of a real system. This system did not exist prior to this research; however, the wavelet transform code and the SNNS tool did exist. The scaling function component and the algorithm to determine when the network is fully trained were created for this research. The wavelet component used in this research is explained in more detail in Section 3.7.2. The neural network is created and modeled using SNNS, a widely employed neural network modeling program.

Real images are used as a workload. Since the images are modeled after one of the eight models the system was trained on, there are no unknown inputs. This allows comparison of the output of the system to the desired output.

### **3.9 Experimental Design**

A full factorial design is run for each neural network topology and data transform for each subband. Therefore, each topology of each subband neural network uses each of

the four post-wavelet data transformations. Each of these training sessions is run at least four times with a different random seed to account for random variations during initializations and weight jogging. Since there are four data transforms, three network topologies, and all variations are run four times with different random seeds,  $4 \times 3 \times 4 = 48$  experiments are conducted to determine the best data transform and network topology for each subband. After training, each of these nine selected subband networks is tested with testing sets. Each subband is considered independently. Additionally, several large networks composed of the smaller subband networks are tested.

The wavelet transform and data scaling function always provide the same output for a given input and do not need to be considered in the factorial design. The order of presentation of patterns to the neural network also does not need consideration since they are randomly selected after each training epoch. The training process runs enough of these randomly shuffled epochs that these variations should be significantly reduced. Additionally, this factor is encompassed by using different random seeds. Ultimately, the mean square error of the training set and the testing sets should become stable as the neural network “learns” from experience and reaches a steady state.

One of the primary goals of this research is to determine if the multiresolution aspect of the wavelet transform is exploitable. To achieve this goal, separate neural networks for each subband are created and the outputs correlated using a larger network. Measurements are taken of both the individual subband networks and the larger correlating networks using testing sets. Since the feedforward process of a neural network is deterministic, multiple runs are not required. The testing sets only need to be run a single time to determine their success rates.

### 3.10 Analyze and Interpret Results

The data gathered consists of 432, four-valued sets, one set for each variation of subband, network topology, data transformation, and random seed. Each set contains the following information:

- Epoch at which the network is fully trained (“trained epoch”)
- Training set MSE
- Testing set MSE
- Success rate

The trained epoch, training set MSE, and testing set MSE data values are not used to gauge the performance of the system, but only to ensure that fully trained, stable networks are considered.

The interaction between factors is not calculated, because this research is concerned only with choosing the best performing network topology and scaling function. An example table used to collect data values is shown in Table 3.1. A table is constructed for each of the data types (trained epoch, trained MSE, test MSE, and success rate). Four tables are created for each subband.

This system is affected by the random initialization state of the neural network and the random jogging that occurs after each training epoch. Also, very small variations in any parameter have an undetermined affect on the output.

### 3.11 Summary

This research creates an image classification system using a wavelet transform, scaling function, and a neural network. Neural networks are used because they are ideal for

solving non-linear problems and other problems in which a formula cannot be determined. Also, prior research efforts have successfully used neural networks to solve difficult problems.

Table 3.1. Example data collection table for success rate for level 3, subband 3HH.

Network Toplogy (hidden1 – hidden2)	Random Seed	Scaling Function Success Rate (%)			
		None	Linear	Log	Both
128 - 32	1	66.372	81.813	80.494	83.288
128 - 32	2	64.833	84.895	83.194	85.501
128 - 32	3	66.372	83.198	81.631	85.55
128 - 32	4	65.544	80.86	83.194	86.207
128 - 64	1	73.669	86.566	82.506	87.727
128 - 64	2	73.077	84.447	82.118	86.004
128 - 64	3	74.219	85.025	82.192	85.71
128 - 64	4	72.598	84.436	82.118	85.734
128 - 128	1	92.725	90.241	78.567	93.768
128 - 128	2	82.467	91.03	77.687	92.061
128 - 128	3	82.398	91.32	75.659	91.535
128 - 128	4	81.604	92.732	77.374	90.667

Ideally, a presented object should be correctly classified regardless of elevation or rotation angle. Using the wavelet transform as a pre-processor, many combinations of network topologies and data rescaling functions are processed to determine the best settings for optimizing the success rate.

#### IV. Analysis and Results

This chapter provides the results of training and testing of the neural networks as described in Chapter 3. Each subband of each level is trained and tested independently. Analysis and conclusions of the results are also presented. Section 4.1 examines the results for the lowest resolution scale, level 3. Section 4.2 examines the findings for level 2. Section 4.3 examines the findings for level 1, the highest resolution scale. Figure 4.1 provides an overview of the wavelet decomposition and associated levels and subbands. Section 4.4 analyzes the results of creating larger networks from the subband networks from multiple levels. Finally, Section 4.5 lists the findings of training the level 3 and level 2 subband networks on a photo training set and testing with a photo testing set.

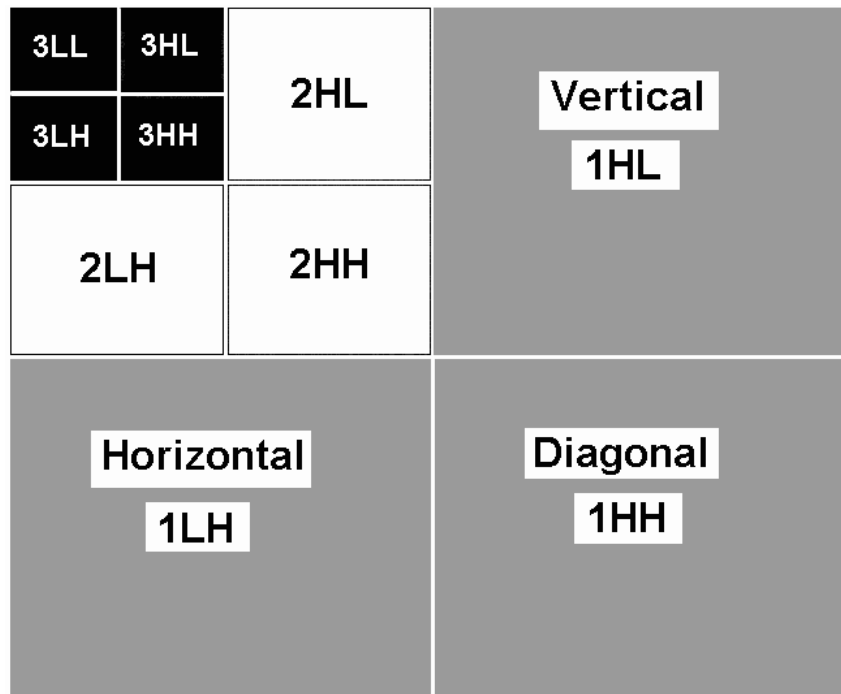


Figure 4.1. Levels and subbands of the wavelet decomposition.

Detailed statistical analysis is not performed since the focus of this research is to discover the merit of the wavelet transform. This is done by examining the success rate and not the interaction between factors. Throughout each section the significant details in graphs and tables are noted.

#### **4.1 Level 3 (Lowest Resolution)**

The level 3 subband networks use the lowest resolution subbands as inputs. The results are obtained from the training and testing of various neural network topologies with the four different data transformations. Each subband is trained and tested independently. Section 3.7.4 contains a more detailed description of this procedure.

Initial training of each network is conducted to 120 epochs which over trains the networks. Overtraining is done to gather statistics that determine the epoch at which the network is fully trained (see Section 3.7.6). Over trained networks are not used to conduct testing. Statistics are only gathered at the point at which the network is fully trained. After conducting the training runs, the following general results in Table 4.1 are obtained. The first column indicates the factor. The first four rows in this column show the level and subband. The next three rows indicate the network topology where the first number represents the resolution of the subband ( $N \times N$  pixels). The second and third numbers indicate the number of nodes in the first and second hidden layers respectively. The final four rows of the table show the results for the four data transforms.

Table 4.1. Level 3 results for each factor.

Factor	Mean (%)	Success Rate (%)		Std Error
		Lower 95%	Upper 95%	
<b>Level, Subband</b>				
3HH	81.69	80.42	82.96	0.6438
3HL	86.67	85.4	87.94	0.6438
3LH	90.71	89.43	91.97	0.6438
3LL	18.65	17.38	19.92	0.6438
<b>Network</b>				
16-128-32	70.59	69.49	71.69	0.5575
16-128-64	69.15	67.05	69.25	0.5575
16-128-128	69.55	68.45	70.65	0.5575
<b>Scaling Function</b>				
Both	71.56	70.3	72.84	0.6438
Linear	74.22	72.95	75.49	0.6438
Log	67.97	66.7	69.24	0.6438
None	63.96	62.69	65.24	0.6438

It is evident that the 3LL subband is an under performer which is consistent with wavelet theory. 3LL is created by executing a low-pass filter on both the rows and columns of the image which results in a non-detailed representation of the image. Basically, no edge information is obtained, only “blobs” of color. The three other subbands’ performance is very close, indicating that information is evenly distributed between these subbands. This coincides with previous research that explored a handwritten number recognition system using wavelets [Corr02]. All three network topologies have nearly the same success rates, therefore, the best success rate among these three networks cannot be determined from the information in Table 4.1. For this reason, the most computationally efficient topology, 128 nodes in the first hidden layer and 32 nodes in the second hidden layer, is chosen. Examining the scaling function

results from Table 4.1 reveals that the results are close, but the linear scaling appears to outperform the other scaling functions. However, when the results are viewed in more detail in Figure 4.2, linear scaling does not outperform log scaling. Figure 4.2 allows comparison of results for the different network topologies, data transformations, and subbands. Note that the log scaling has a lower variance in 3 of the 4 subbands. Since both the log and linear scaling functions have nearly the same performance, the log scaling function generally exhibits a lower variance, and since the log function closely matches the natural exponential distribution of the wavelet coefficient values, the log data transform is selected. The distribution of the wavelet coefficients is explained in Section 2.2.2.2,.

Now that the best data transform and best network topology for level 3 are decided, an average performer is chosen as explained in Section 3.7.5. This is done by running the selected choices with several random seeds. Table 4.2 shows the results with the chosen selections highlighted. “Trained epoch,” the epoch at which the network is fully trained, is included because this indicates where training for each network, using the log scaling function, should cease. The neural networks are saved at the next highest multiple of 5. For example, if the trained epoch is 113, then it is rounded up to 115. This is permissible since the networks are acceptably stable at this point and because only a small number of additional training epochs are run. Additionally, the MSE does not climb which would indicate an over trained network. Figure 4.3 shows the results for the 3HH subband network and the MSE for training and testing. The success rate is also shown and the trained epoch point is indicated. Appendix A contains the other subbands’ results which are similar in appearance.



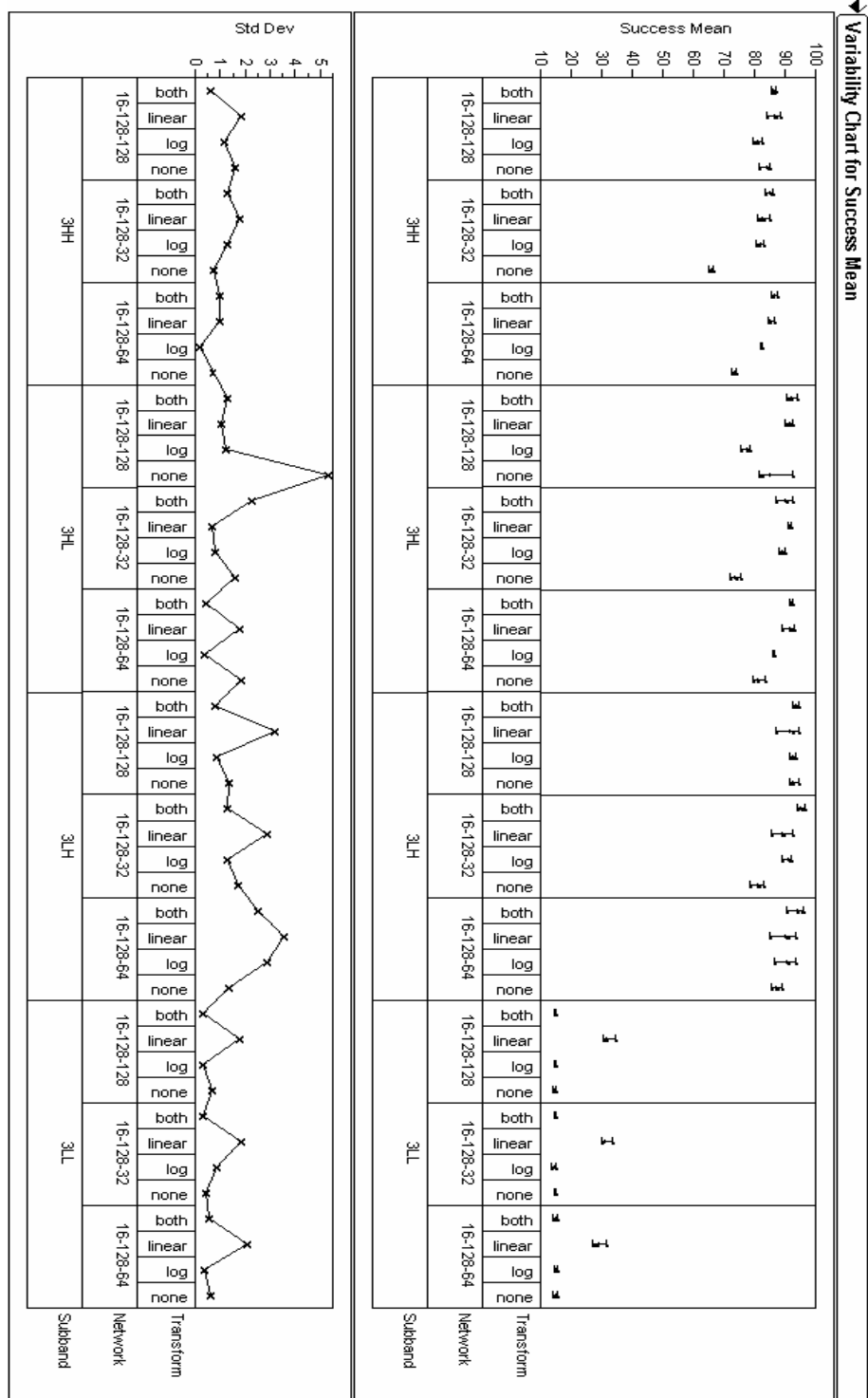


Figure 4.2. Level 3 variability chart for the CAD set success rate.

Table 4.2. Results of level 3 ran with random seeds.

Subband	Network	Scaling	Trained	Success (%)
3HH	16-128-32	log	113	84.456
3HH	16-128-32	log	131	84.408
3HH	16-128-32	log	109	80.37
3HH	16-128-32	log	110	86.074
3HL	16-128-32	log	74	88.139
3HL	16-128-32	log	70	86.985
3HL	16-128-32	log	100	91.083
3HL	16-128-32	log	72	86.498
3LH	16-128-32	log	74	93.853
3LH	16-128-32	log	89	94.161
3LH	16-128-32	log	80	95.175
3LH	16-128-32	log	73	93.986

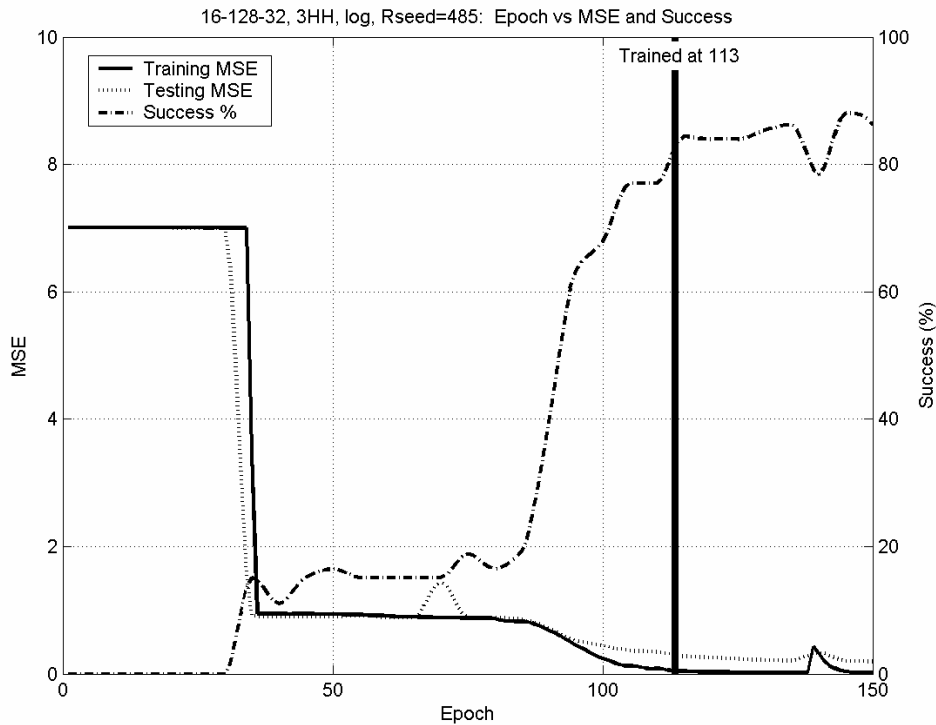


Figure 4.3. Level 3, HH subband trained epoch.

After all three subband networks are trained (i.e. 3LH, 3HL, 3HH), larger networks are composed of these smaller subband networks. The purpose of the large networks is to correlate the outputs of the smaller networks. Three different configurations are tested. One configuration, pictured in Figure 4.4, correlates the outputs from each subband network. This is not a fully-connected architecture. The other two “glue” or level networks are fully connected. One of these networks features an eight node hidden layer interposed between the outputs of the subband neural networks and the final outputs of the system. The other fully-connected network does not have a hidden layer.

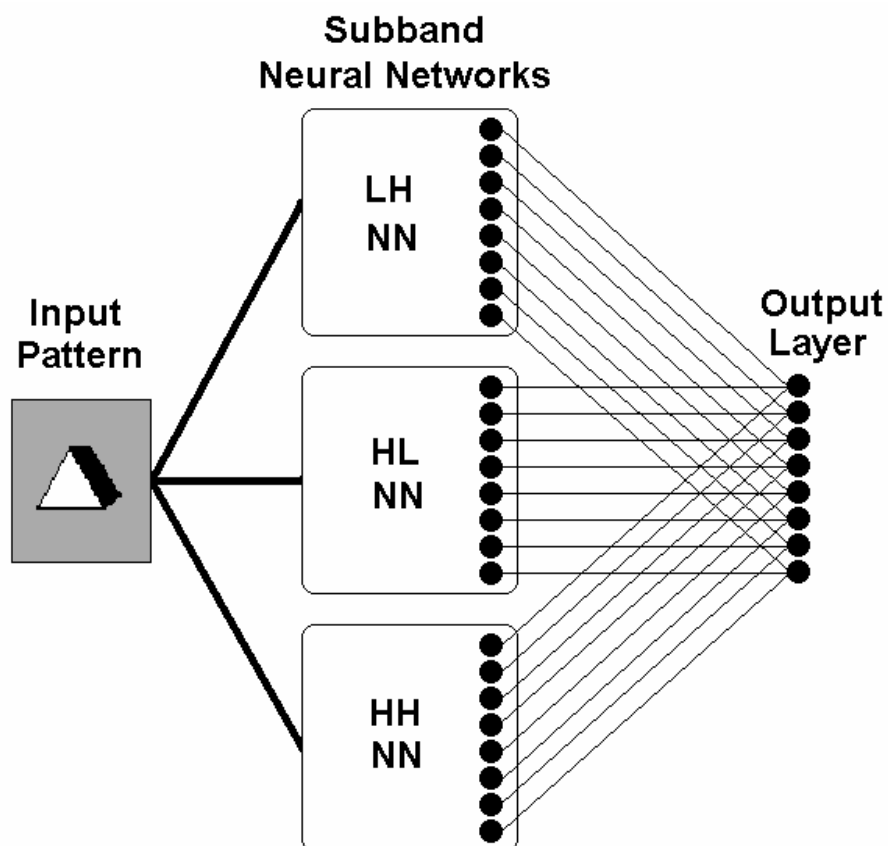


Figure 4.4. Large correlating network connecting the subband networks for a level.

Since the inputs to these level networks are relatively simple given the high success rate of the three input subband networks, the level network trains quickly, as seen in Figures 4.5 and 4.6. This is especially true for the network without hidden nodes which trains in only two epochs. The graph titles indicate the number of inputs (3 subband networks x 8 outputs each = 24 inputs), level, and testing set. Mean square error (MSE) is shown to indicate where the network has become stable. These three networks are tested with each of the four testing sets. For brevity, only the CAD testing set is analyzed in this discussion. However, this demonstrates the analysis procedure. Section 4.4 contains results for all test sets.

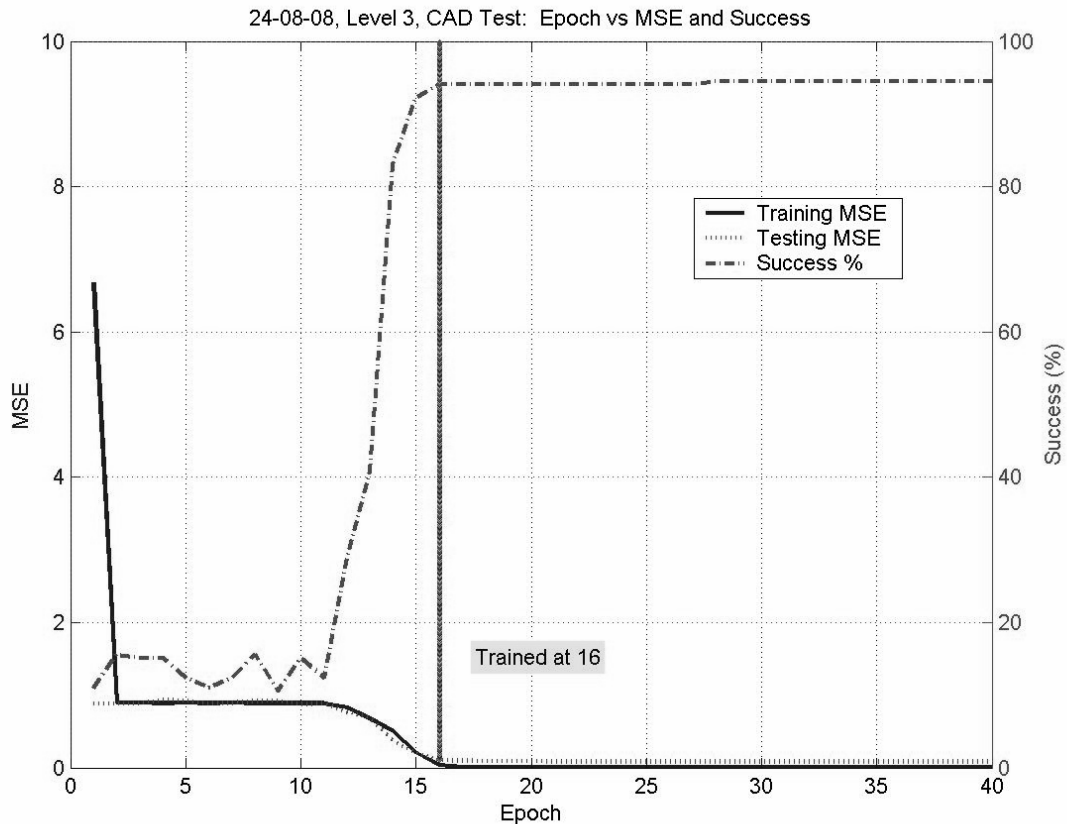


Figure 4.5. Results of the CAD testing set on the level 3 “glue” fully connected network with hidden nodes.

From visually inspecting Figure 4.5, the graph of the success rate of the CAD testing set, the network is fully trained after 16 epochs. The success rate at this point is 94.5% (206 of 218 patterns) which is slightly better than any of the individual subband networks. This is expected as hidden nodes increase performance at the expense of stability and training time.

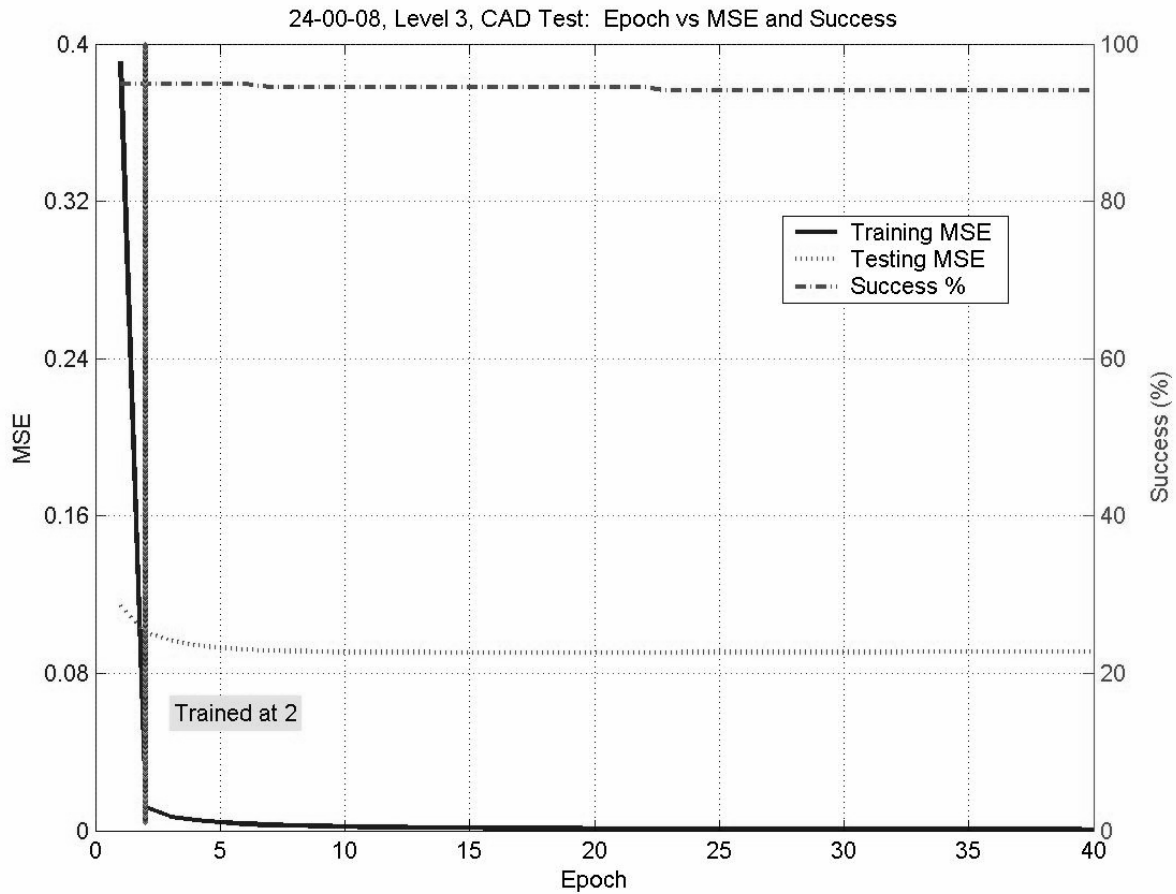


Figure 4.6. Results of the CAD testing set on the level 3 “glue” fully connected network without hidden nodes.

From inspection of Figure 4.6, the network without hidden nodes is fully trained in only two epochs, and achieves a success rate of 94.0% (205 of 218 patterns). This is only one pattern worse than when using the hidden layer nodes. This agrees with Section

2.1.4 which states that hidden nodes help the network solve more complex problems at the expense of training time.

The graph for the correlated network results shown in Figure 4.7 is very similar to the fully-connected network results from Figure 4.6. The success rate is 94.0%.

Typically, across all level networks tested, the correlating networks yield success rates almost exactly the same as the no-node, fully connected networks, but require a few additional training epochs.

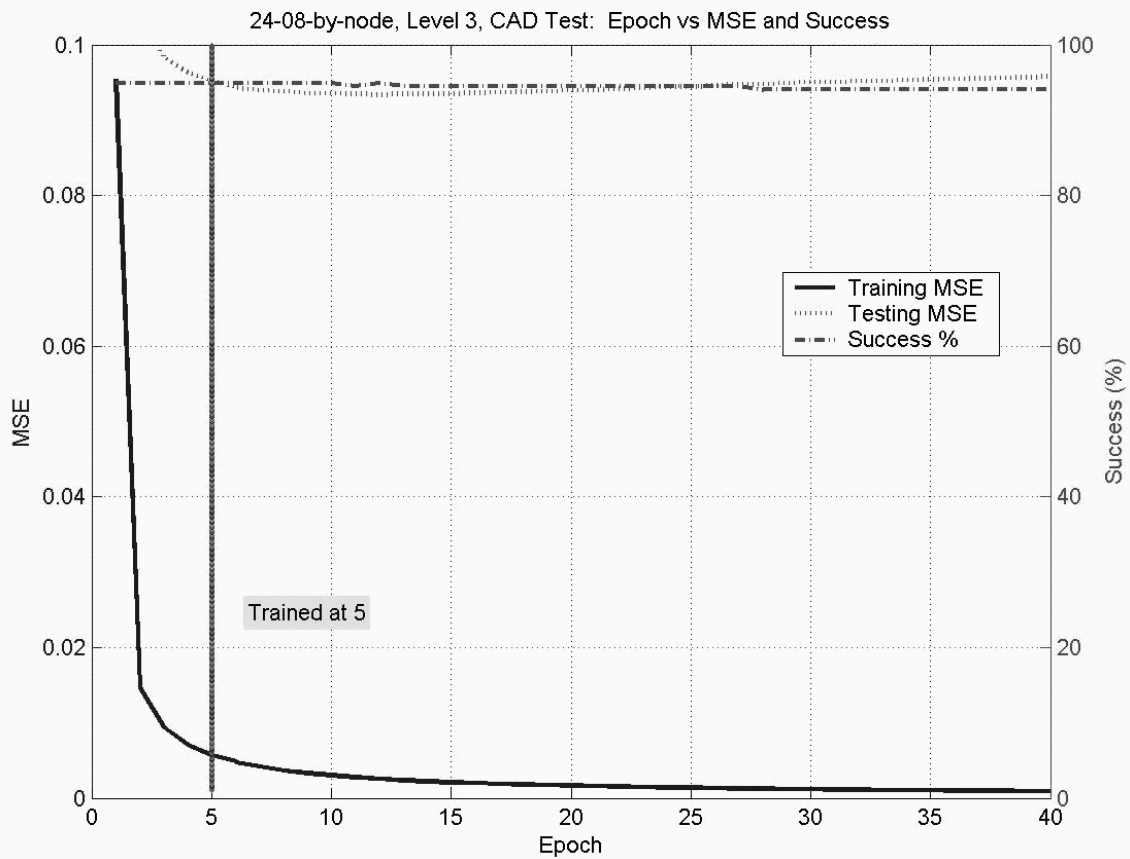


Figure 4.7. Results of the CAD testing set on the level 3 correlating network.

The patterns that are falsely identified are examined for all three networks using a confusion matrix and a script that lists the name of the misclassified patterns. Twelve (12) of the same patterns were misidentified by all three networks. However, examining these images is inconclusive; there is no visually obvious reason why these patterns should be incorrectly identified. For example, in one case the half-circle (“moon”) object was misidentified as the triangular object. In other cases, the rectangular object was mistaken for the moon object, the triangular object, and the long tunnel shaped object.

Table 4.3 shows a summary of the success rates for all subband and level networks tested with all four testing sets. Success rates are measured at epoch 40. In almost all cases, the level networks have a higher success rate than the individual subband networks with the exception of the 3HH subband network when tested with the good photos set. The reasons for this are unknown, but across all subband and level networks tested, 33.3% is the highest success rate achieved on the good photos set.

Table 4.3. Subband and level network success rates (%) for level 3.

<b>Network</b>	<b>Connection</b>	<b>CAD Train</b>	<b>CAD Test</b>	<b>All Photos</b>	<b>Good</b>
<b>3HH</b>	<b>Full</b>	97.2	84.4	14.3	6.1
<b>3HL</b>	<b>Full</b>	94	83.9	11.1	33.3
<b>3LH</b>	<b>Full</b>	98.2	93.1	15.1	12.1
<b>Level 3</b>	<b>Full</b>	100	94	16.7	21.2
<b>Level 3</b>	<b>Full w/Hidden</b>	100	94.5	15.1	24.2
<b>Level 3</b>	<b>Correlated Outputs</b>	100	94	16.5	21.2

### 4.3 Level 2

The creation and training of the level 2 subbands follows the same process as is performed for the level 3 networks. The networks are trained and statistics gathered at the epoch at which the network is fully trained. Initial results are displayed in Table 4.4.

Table 4.4. Level 2 results for each factor.

Factor	Mean (%)	Success Mean (%)		Std Error
		Lower 95%	Upper 95%	
<b>Level, Subband</b>				
2HH	77.13	76.53	78.41	0.4754
2HL	85.34	85.06	86.83	0.449
2LH	87	86.39	88.25	0.4723
<b>Network</b>				
32-128-32	83.14	82.48	84.34	0.4723
32-128-64	82.32	81.7	83.51	0.46
32-128-128	84.36	83.81	85.63	0.4629
<b>Scaling Function</b>				
Both	85.92	84.66	86.83	0.5503
Linear	87.6	86.48	88.72	0.57
Log	79.42	78.25	80.1	0.4701
None	81.8	80.67	82.92	0.57

Overall, the results for level 2 are very much like those for level 3. As before, the network topology makes little difference so the most computationally efficient network is chosen. Examining the data transform results seem to initially reveal that the linear transform should be chosen and, contrary to the level 3 results, the log transform is now the worst performer. For the sake of brevity and to show more clearly the regions of interest of the variability chart, only the 128-32 network topologies from the variability chart are shown in Figure 4.8. Appendix B contains all of the variability charts for the level 2 tests.

Figure 4.8 also displays a graph that shows the epoch at which the network is fully trained. This information is not presented for the level 3 networks in Section 4.1, but can be found in Appendix A. It is included here to show how the log scaling is not



only the worst performer for success rate, but also to demonstrate the high degree of variability for the trained epoch.

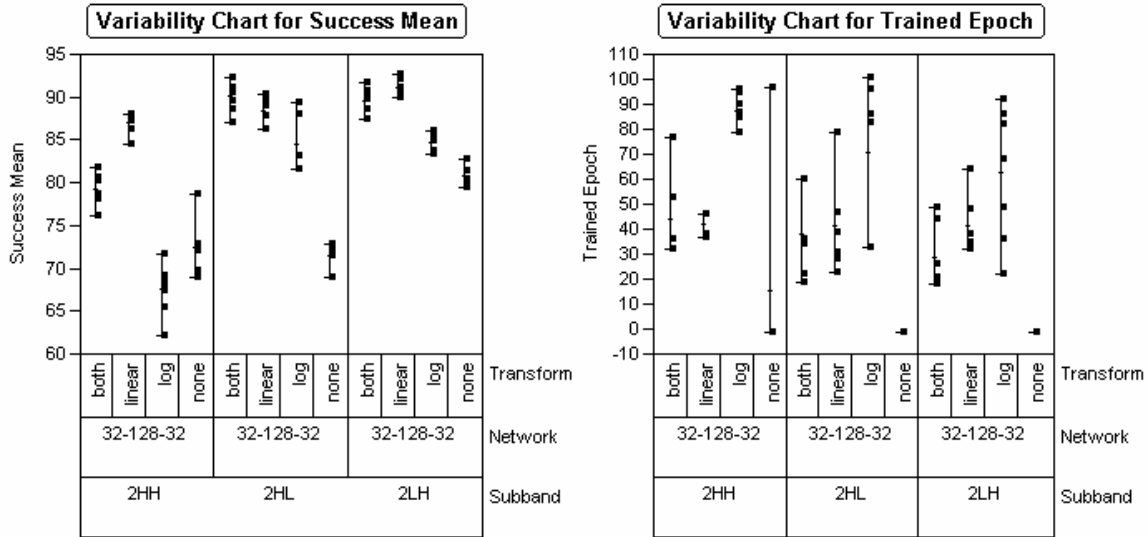


Figure 4.8. Level 2 variability chart for the CAD set success rate and trained epoch.

This variability is a non-favorable characteristic. The linear data scaling function is chosen since it typically has lower variability and a generally higher success rate than the other transforms.

The chosen selections are run with several random seeds to determine an average performer as explained in Section 3.7.5. The network topology (128-32), scaling function (linear), and random seed have now been decided and the final level 2 networks are trained. The chosen networks are highlighted in Table 4.5.

From this point, the same process is followed as in Section 4.1. Once again, three glue networks are created that use the outputs of the subband networks as inputs. The results of the CAD testing set are covered in this section. Appendix B contains more detailed variability and subband results. As before, the network without hidden nodes

trains much faster than the network containing hidden nodes. The correlating network takes five epochs to train, but boasts the highest success rate on the

Table 4.5. Results of level 2 ran with random seeds.

Subband	Network	Scaling	Trained	Success (%)
2HH	32-128-32	linear	38	85.458
2HH	32-128-32	linear	37	84.604
2HH	32-128-32	linear	31	87.553
2HH	32-128-32	linear	24	87.377
2HH	32-128-32	linear	32	83.784
2HL	32-128-32	linear	38	89.757
2HL	32-128-32	linear	36	87.652
2HL	32-128-32	linear	41	86.387
2HL	32-128-32	linear	54	88.341
2HL	32-128-32	linear	32	90.97
2LH	32-128-32	linear	38	90.716
2LH	32-128-32	linear	52	89.269
2LH	32-128-32	linear	32	92.067
2LH	32-128-32	linear	62	87.946
2LH	32-128-32	linear	32	88.542

CAD testing set. It should be noted that the same three shapes were misidentified by all three level networks. Identifying the misidentified objects yields no clues as to why they are misclassified. For instance, the arch and the moon objects were misclassified as the rectangle object. Table 4.6 lists the results for the level 2 subband and level networks.

Table 4.6. Subband and level network success rates (%) for level 2.

Network	Connection	CAD Train	CAD Test	All Photos	Good Photos
2HH	Full	95.5	87.2	17.0	18.2
2HL	Full	94.1	85.8	9.0	21.2
2LH	Full	96.6	89.4	12.5	9.1
Level 2	Full	100.0	97.7	13.8	15.2
Level 2	Full w/Hidden	100.0	97.7	14.6	15.2
Level 2	Correlated Outputs	100.0	98.2	13.8	15.2

### 4.3 Level 1 (Highest Resolution)

The same procedure that is used for levels 2 and 3, is also used for level 1. The most significant difference is the amount of time that it takes to train these larger subband networks. These networks have 4096 inputs (64 x 64 pixels) and take over 16 times long to train than the level 3 subband networks and over four times longer than the level 2 networks. For this reason, after choosing the best performing network topology and data transform, only a single additional run is performed. The results of the pilot studies and this additional run are used to select the average performer as explained in Section 3.7.5.

Table 4.7 reveals the general results for the level 1 subband networks. Additional network topologies are trained and tested under the hypothesis that additional hidden nodes may be necessary due to the larger number of inputs. Recall that the level 3 subband neural networks only have 256 inputs, level 2 networks have 1024 inputs, and level 1 networks have 4096 inputs. However, this hypothesis is not fully tested in this research due to the amount of computational time required to run these tests. Tests not using any scaling function are not run on the 256 hidden node networks due to time constraints and the poor performance in the previous levels when no scaling function is used.

From the results presented in Table 4.7, it initially appears that the 128-32 network should be chosen as well as linear scaling. These preliminary results are verified against the abbreviated variability chart shown in Figure 4.9. The last three network topologies use 256 nodes in their first hidden layer. Log scaling is run first on these larger networks and the results compared to the smaller networks that use 128 nodes in the first hidden layer. No further benefit is gained from these larger networks.

Table 4.7. Level 1 Results for each factor.

Factor	Mean (%)	Success Rate (%)		Std Error
		Lower 95%	Upper 95%	
<b>Level, Subband</b>				
1HH	62.53	65.8	68.48	0.6702
1HL	80.15	78.78	81.47	0.6732
1LH	79.3	81.45	84	0.6366
<b>Network</b>				
64-128-32	75.48	74.21	77.02	0.7011
64-128-64	76.04	74.73	77.36	0.6548
64-128-128	77.53	76.22	78.84	0.6548
64-256-32 (log)	65.06	77	82.93	1.482
64-256-64 (log)	67.67	73.24	77.95	1.1767
64-256-256 (log)	67.3	72.87	77.58	1.1767
<b>Scaling Function</b>				
Both	79.4	77.87	80.57	0.6732
Linear	88.31	85.63	88.94	0.8259
Log	62.74	62.44	64.54	0.525

Also, due to the increased computational requirements of these larger networks no further tests are performed using these networks. Appendix C contains the variability chart displaying this information as well as more results from the level 1 tests.

The results pictured in Table 4.7, are used to choose the random seed to use for training for each subband in level 1. Due to time constraints, only three iterations for each chosen network and data transform are run. The success rates for each subband are very close although the epoch at which the network is fully trained varies widely. The randomness of the initialization state and the random jogging that occurs during training explains this variance. This is analogous to rolling a ball down a hill. The ball hits different bumps along the way, but eventually settles at the bottom of the hill.

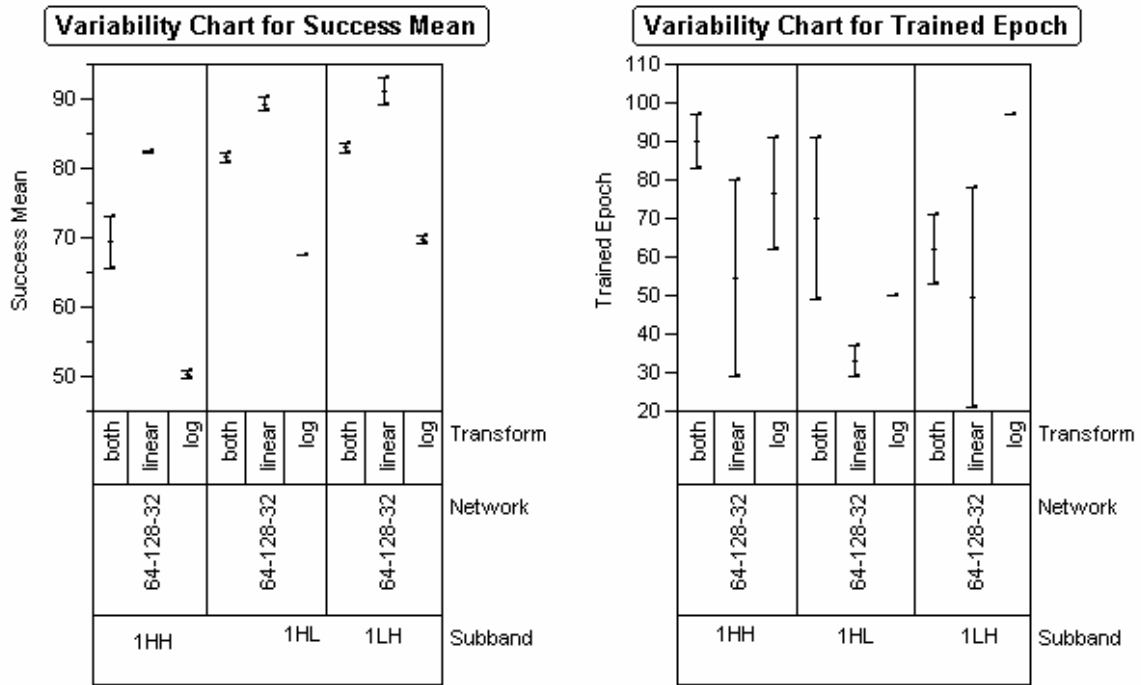


Figure 4.9. Level 1 variability chart for the CAD set success rate and trained epoch.

However, this variance does not impact the results since only fully trained networks are used for testing.

Based on the results from Table 4.7 and, more importantly, from the variability charts, the 128-32 network topology is selected. The linear data transform is selected based on its strong performance shown in Figure 4.9.

The selected topology and data transform are run once more with a different random seed and all the runs for this topology and data transform are compared in Table 4.8. The average performer is chosen and the network fully trained. Section 3.7.5 explains the reasoning for choosing the average performer.

The individual, trained subband networks are used to create larger networks for level 1. Table 4.9 lists the results for level 1. All three networks have consistent performance and all three misclassify the same seven patterns for the CAD testing set.

Table 4.8. Results of level 1 subband networks ran with random seeds.

Subband	Network	Scaling	Trained	Success (%)
1HH	64-128-32	linear	29	82.217
1HH	64-128-32	linear	80	82.626
1HH	64-128-32	linear	77	82.147
1HL	64-128-32	linear	29	90.309
1HL	64-128-32	linear	37	88.282
1HL	64-128-32	linear	62	90.824
1LH	64-128-32	linear	21	92.936
1LH	64-128-32	linear	78	89.113
1LH	64-128-32	linear	39	91.156

The training times are consistent with the level 2 and level 3 networks. The fully connected, hidden node network takes the longest time to train and the fully-connected no-hidden network takes two epochs. The correlating networks take slightly longer than the no-hidden network at three training epochs.

Table 4.9. Subband networks' and level networks' success rates (%) for level 1.

Network	Connection	CAD Train	CAD Test	All Photos	Good Photos
1HH	Full	97.7	84.9	12.5	9.1
1HL	Full	99.1	89.0	9.0	0
1LH	Full	96.4	89.9	17.5	24.2
Level 1	Full	100.0	96.3	12.5	9.1
Level 1	Full w/Hidden	100.0	95.9	13.0	9.1
Level 1	Correlated Outputs	100.0	96.3	12.2	9.1

#### 4.4 Multi-Level Networks.

As described in Sections 2.1.1 and 2.1.10, neural networks are massively parallel, especially the architectures used in this research. A VLSI hardware implementation could process every subband in each level simultaneously. The level 3 subband networks would complete first, followed by level 2, and finally by level 1. The size of the input layer size and first hidden layer greatly outnumber the other nodes in the networks. For this reason, the computational complexity is largely the product of the number of nodes in these two layers. The level 3 subband networks executes in about half the time required for the level 2 networks which requires about half the time required for the large level 1 networks. The glue networks are small in comparison and computational time is insignificant when compared with any of the subband networks. Since almost all of the computational time is spent processing the subband networks, use of the correlating networks is almost a given since they always provide an increase in success rate on a per-level as indicated in Figure 4.10. All the graphs shown in this section use only the fully-connected, level networks without hidden nodes since they consistently provide the best success rate performance.

Figure 4.10 displays the results for the CAD testing set for all the subband and level networks. Note that the network trains only on the CAD training set and has never been presented with any images from the CAD testing set. The computational time requirement is smallest at the top of the graph (subband networks in level 3) and increases towards the bottom of the graph. The level networks require very little additional computational resources, but cannot be computed until all input subband

networks complete. This also applies to the network that correlates all of the level 2 and 3 networks and for the network that correlates all nine subbands.

Table 4.10 reveals a summary of all the sets' performances on the different subband and level networks. Both photo sets show poor performance. The photo set consisting of all photos is generally consistent at around 12% - 16%. The good photo set shows much more variance, however, the HL subbands consistently outperform all other networks. The other two subbands, HH and LH, drag the performance of the level networks down; they do not even achieve the modest success of the HL networks. These trends are easily seen in Figure 4.11.

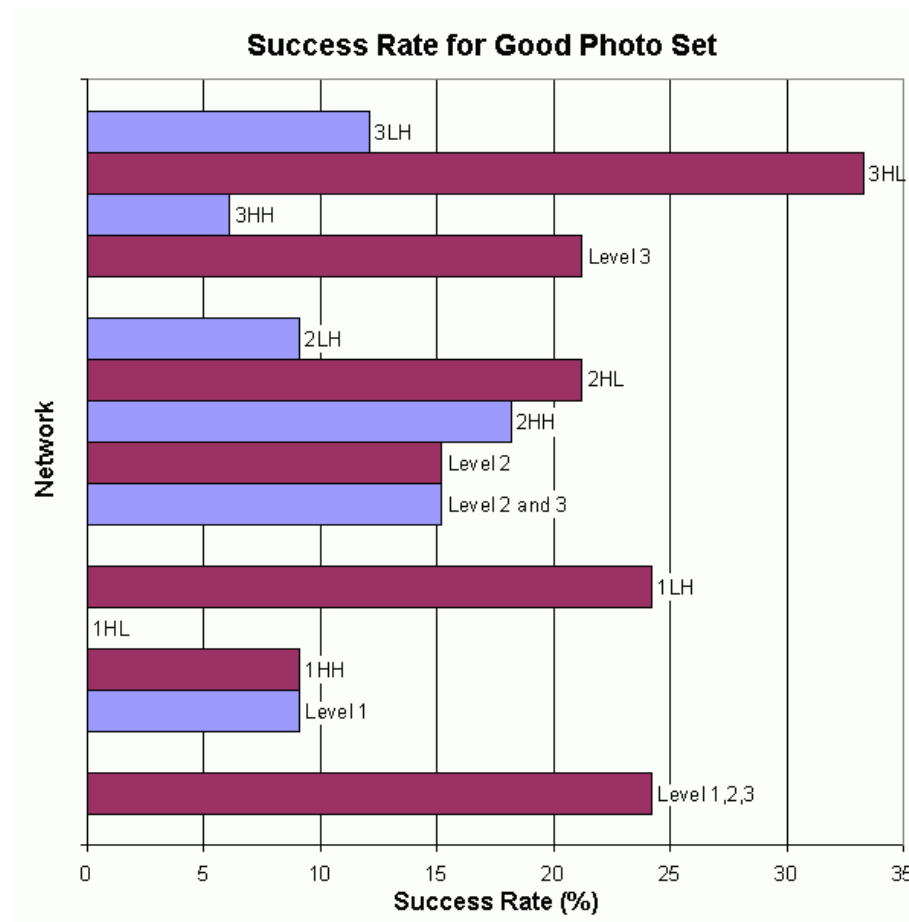


Figure 4.10. Success rates for subband and level networks tested on the CAD testing set.



Table 4.10. Overall results (%) for all networks.

Network	CAD Train	CAD Test	All Photos	Good Photos
3HH	97.2	84.4	14.3	6.1
3HL	94.0	83.9	11.1	33.3
3LH	98.2	93.1	15.1	12.1
Level 3	100.0	94.0	16.7	21.2
2HH	95.5	87.2	17.0	18.2
2HL	94.1	85.8	9.0	21.2
2LH	96.6	89.4	12.5	9.1
Level 2	100.0	97.7	13.8	15.2
Level 2,3	100.0	97.3	15.4	15.2
1HH	97.7	84.9	12.5	9.1
1HL	99.1	89.0	9.0	0
1LH	96.4	89.9	17.5	24.2
Level 1	100.0	96.3	12.5	9.1
Level 1,2,3	100.0	97.7	14.6	24.2

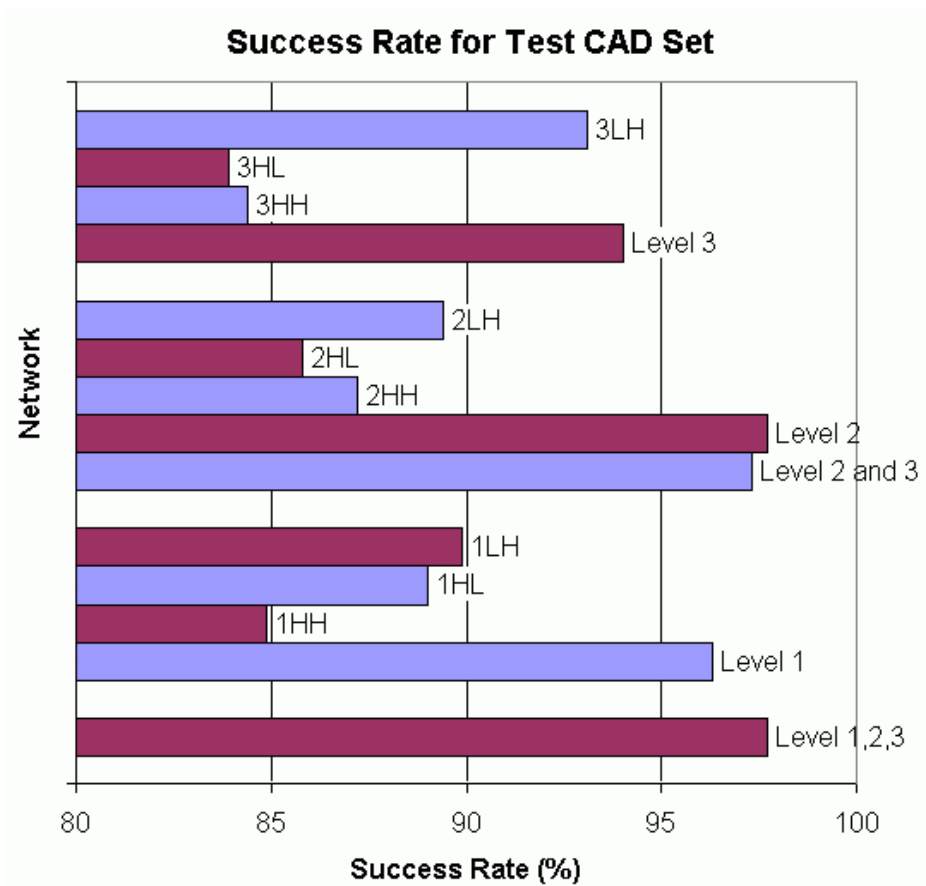


Figure 4.11. Success rates for subband and level networks tested on the good photo set.

## 4.5 Verification Tests

In addition to the tests discussed in the previous four sections, the selected networks and data transforms are trained on the photo set. Following the same procedure as the CAD training, 90% of the photo patterns are used for training and the remaining 10% used as a photo testing set. Once training completes, the trained neural networks are tested on the photo test set and on a CAD testing set. Ten (10) runs with different random seeds were executed for the photo training and testing set and the results averaged. The results of these tests are shown in Table 4.11.

Table 4.11. Success rates (%) of photo training vs. CAD model training.

Network	Trained on CAD Train Set				Trained on Photo
	CAD Train	CAD Test	All Photos	Good Photos	Test Photos
3HH	97.2	84.4	14.3	6.1	22.9
3HL	94.0	83.9	11.1	33.3	35.2
3LH	98.2	93.1	15.1	12.1	31
2HH	95.5	87.2	17.0	18.2	16.8
2HL	94.1	85.8	9.0	21.2	36.6
2LH	96.6	89.4	12.5	9.1	31.4

In most cases, the networks have a much higher success rate when only training on the photo models. This could be the result of the neural network keying on particular features since only 337 images are used for training. It is interesting, but not unexpected, that corresponding subbands in different levels demonstrate about the same amount of success for each of the photo test sets (each column) whether the system is trained on CAD images or photo images. This follows the wavelet coefficient persistence property discussed in Section 2.2.2.2. The HH subbands under-perform the other two subbands when trained on the photo set. The HL subbands have the highest success rates.

A very large neural network is used to test the performance when using unmodified, spatial images. The network is necessarily large so that it can process the entire image at once. Therefore, the input layer consists of  $128 \times 128 = 16,384$  nodes. The first hidden layer consists of 512 nodes and the second layer contains 128 nodes. The output layer, used for classification of the objects, is the same as all networks used in this research; it contains eight nodes. Due to its immense size, this neural network takes an extremely long time to process the input patterns when compared to all the other networks used in this research. Calculating the number of multiply and accumulates required for the networks used in this research generates the following information in Table 4.12.

Table 4.12. Computational requirements (multiply accumulates) of neural networks used in this research.

Network	Nodes (input, hidden)	Feedforward	Back-Propagation
16-128-32	256 - 128 - 32 - 8	37,120	74,016
32-128-32	1024 - 128 - 32 - 8	135,424	270,624
64-128-32	4096 - 128 - 32 - 8	528,640	1,057,056
Glue (1 level)	24 - 8	192	216
Glue (3 levels)	72 - 8	576	648
Spatial	16,384 - 512 - 128 - 8	8,455,168	16,909,440

One of the early pilot studies yields the results shown in Figure 4.12. Figure 4.12 shows that classification of unprocessed images using the spatial network only achieved a success rate of about 16%. Additionally, Table 4.12 reveals that this network requires over 228 times the computations as the level 3 subband networks. Pre-processing images using the wavelet transform and the scaling transform achieved success rates as high as 93% for individual subbands and as high as 97% for the level networks. These results

suggest that the wavelet transform has significant merit as a pre-processing for image classification using a neural network. However, due to the huge computational requirements, the spatial network test was only run one time.

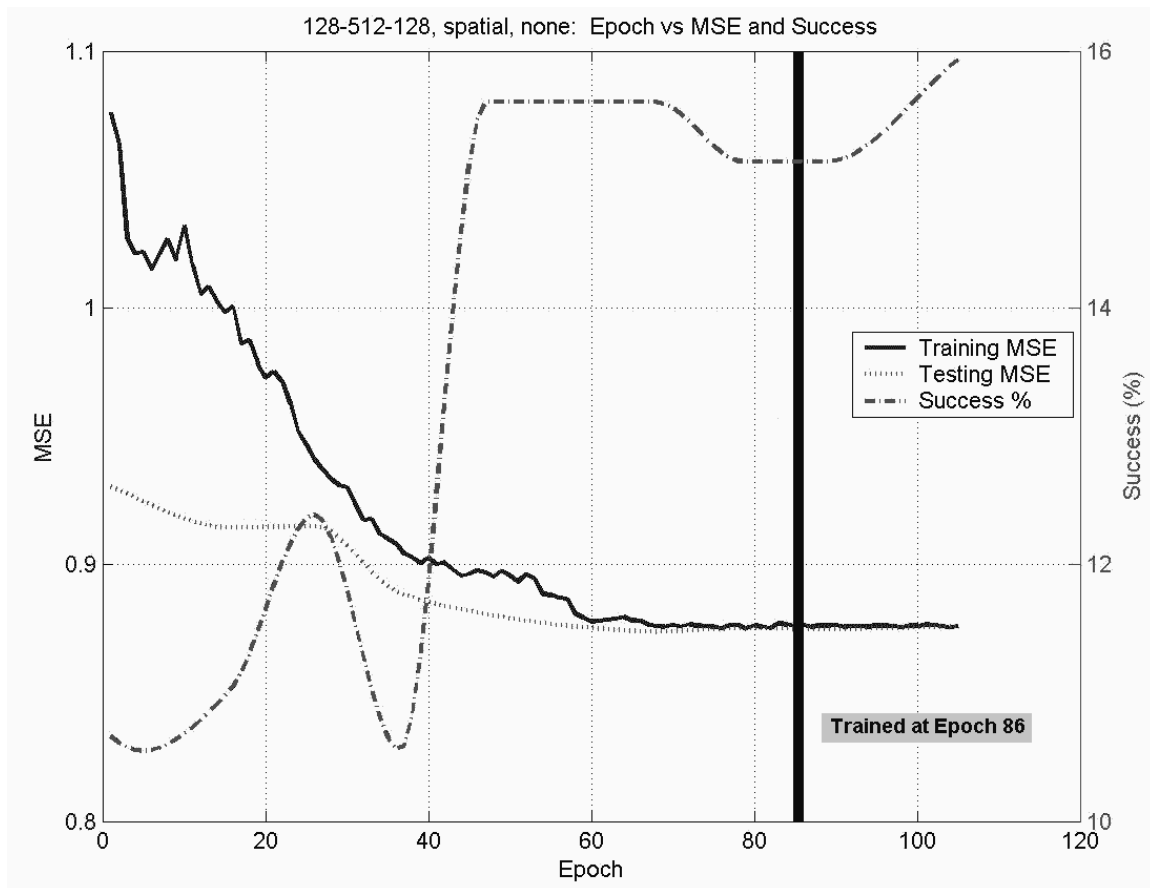


Figure 4.12. Results of processing unmodified spatial images on a single neural network

## 4.6 Summary

This chapter presents the results of determining the best topology and scaling function, training the optimum networks using the selected scaling function, and final testing of the individual subband networks and the larger level networks. The following summarizes the conclusions reached in this chapter:

1. This system performs well when using the CAD modeled data, but poorly on photographed models.
2. The use of 128 nodes in the first hidden layer and 32 nodes in the second hidden layer prove successful for all levels of the wavelet decomposition. Using more hidden nodes does not significantly increase the success rate.
3. Using a scaling function, especially linear rescaling, after the wavelet transform increases the success rate of the neural networks.
4. Correlating the outputs of the individual subbands neural networks is a computationally inexpensive way of increasing.
5. The wavelet transform drastically increases the success rate for object classification when compared to system without using any pre-processing.

## **V. Conclusions and Recommendations**

### **5.1 Summary**

The objective of this research was the exploration of the benefits of the wavelet transform as a pre-processor for object classification by a neural network. The following goals outlined in Chapter 3 and 4 accomplish this objective.

1. Determine if the wavelet transformation is an efficient time and space pre-processor to a neural network for object recognition.
2. Determine which scaling function, if any, increases the success rate of the neural network.
3. Determine which neural network topology proves the most successful.
4. Determine if the multiresolution aspect of the wavelet transform can be utilized to increase the success rate by processing each subband independently and then correlating the results.

These goals are covered in more depth in the following sections.

#### **5.1.1 The Wavelet Transform as a Pre-Processor.**

The wavelet transform is an effective data pre-processor as evidenced in Section 4.5. When using only the CAD model sets, success rates rose from approximately 16% when processing images without the wavelet transform to over 97% when using the wavelet transform. Individual subbands usually had performances around 90% on CAD sets and were never lower than 83.9%. If the subband networks are used to form a larger correlating network, the success rate increases to 97.7%.

However, a conclusive comparison of the performance of neural networks using the wavelet transform to networks processing non-transformed, spatial images is premature. More variations should be run on the large spatial network to validate these findings.

### **5.1.2 Scaling Function.**

The post-wavelet scaling function proves beneficial to the performance of the neural networks. Although the log scaling function was chosen to rescale the wavelet coefficients for the level 3 subband networks, the linear scaling function also could have been chosen. The level 2 and level 1 subband networks use linear scaling. Although this scaling function does not match the natural exponential distribution of the wavelet coefficients, it proved highly successful. This is most likely due to the fact that linearly scaling the coefficient values only preserves the large coefficients corresponding to a sharp change, such as an edge. The other coefficients are effectively reduced to nearly zero. Linearly scaled wavelet coefficients seem to provide a global approach for extracting geometric point primitives from the images.

### **5.1.3 Network Topology.**

The networks consisting of 128 nodes in the first hidden layer and 32 nodes in the second hidden layer demonstrate a success rate very near that of networks with considerably more hidden nodes. Besides being more computationally efficient, these smaller networks are more stable according to neural network theory. Typically, the subband networks train in less than 100 epochs.

#### **5.1.4 Multi-resolution Wavelet Transform.**

The resolution pyramid of the wavelet transform provides a computationally efficient method of lessening the computational burden while providing an effective pre-processor for the neural networks in this research. The three subband neural networks can be parallelized and followed with a small correlating network without significantly increasing computational time over a single subband neural network. If a higher success rate is required, multiple levels of subbands can be processed in parallel without appreciably increasing the computational time over the largest of the input subband networks

### **5.2 Recommendations for Future Research**

This system shows strong performance when tested and trained using the CAD model sets. This section provides some insights into increasing the performance of this system on photographed models or exploring different aspects of the system. Some of these suggestions would be easy to implement while others may require a significant amount of work.

#### **5.2.1 Wire-frame Models.**

Compare the performance of the system described in this research against wire-frame images. Create a spatial pattern set consisting of one-bit, occluded wire-frame models. Rescale the images to obtain the same scale resolutions used in this research. Train and test the neural networks on the spatial sets and on the CAD sets using the same



network configurations. Comparing the performance of the networks using the two sets of data should determine if the wavelet transform is acting only as an edge filtering system or if the wavelet transform also provides some additional information.

### **5.2.2 Additional Training.**

Train the system with more models using finer rotations and elevation angles. Also, create pattern sets using variations in light sources. Model the objects so that shadows are projected. Use different background colors and gradients in the background. Add Gaussian noise to all training sets. All of these make the synthesized images much more real-like and provides many more samples for training.

### **5.2.3 Data Transform.**

Experiment with quantization and thresholding of the scaling functions. This is likely to have the same effect as adjusting parameters for edge detection filters, but may help to overcome noise in the photographed model sets.

### **5.2.4 Scale, Rotation, and Translational Invariant Transform.**

Combine the wavelet transform with scale, rotational, and translationally invariant transforms to provide a representation that is less view dependent to the neural network. By its very nature, the wavelet transform seems ill suited to generalizing rotated objects since it filters the horizontal, vertical, and diagonal directions separately. Any rotation would cause wavelet coefficients to “jump” between directional subbands.

### **5.2.5 Rotation about Axes.**

This research only rotates objects about one axis. If the neural network is forming a good generalized model of the object, then it should be at least somewhat invariant to rotation about different axes. Tests involving rotation about different axes may provide some insights to the internal representation of the models in the neural network.

## **5.3 Final Thoughts**

The object recognition system used in this research performed well on the CAD modeled images. Although success against real photographed models was poor, the use of the wavelet transform as a pre-processor to a neural network proved very successful. Additionally, the wavelet transform provided computationally efficient, multi-resolution scales that could be processed independently by subband neural networks. These individual networks can be used to form larger networks that provide a higher success rate with less variance. If this system is implemented in parallelized hardware, very little additional computational time is required.

Appendix A: Results for Level 3

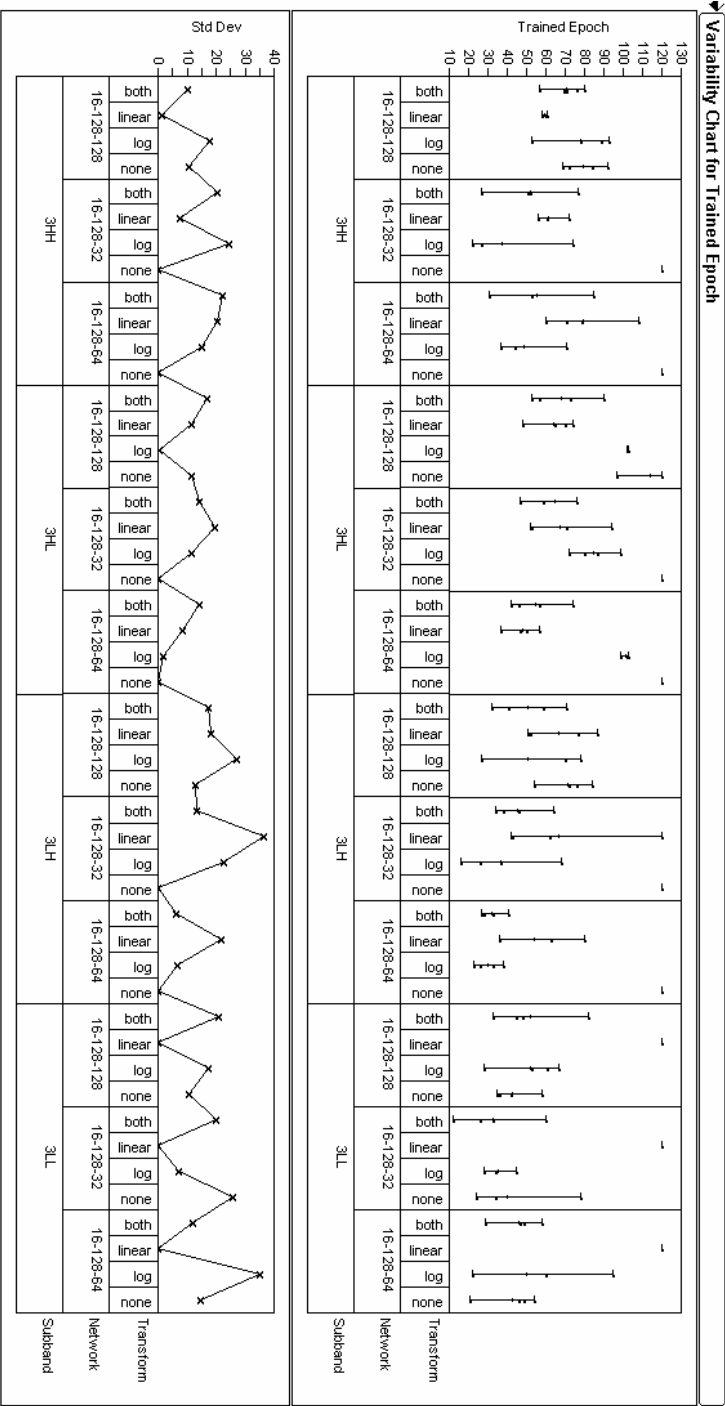


Figure A.1. Level 3 variability chart for the CAD set trained epoch.

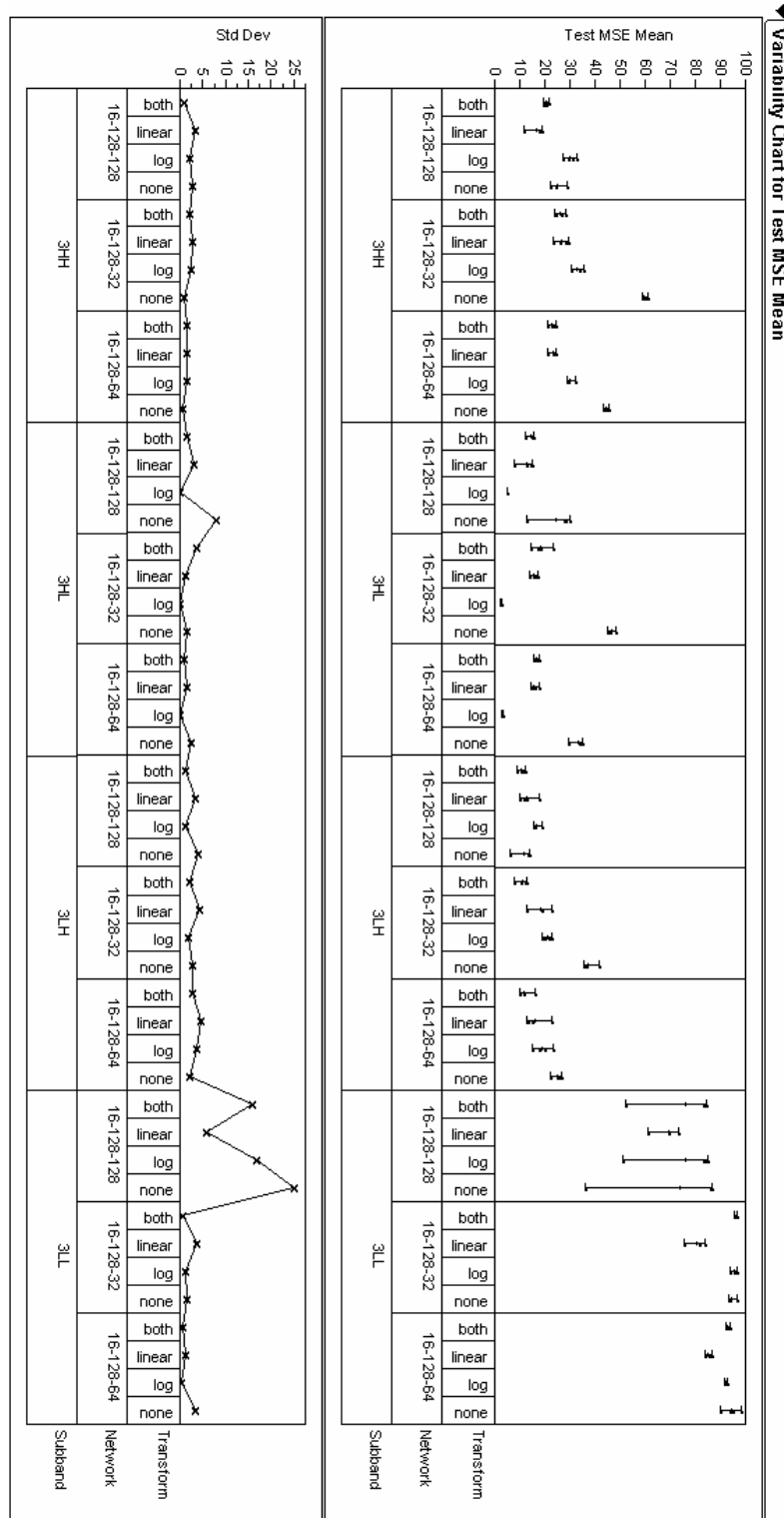


Figure A.2. Level 3 variability chart for the CAD training set mean square error.

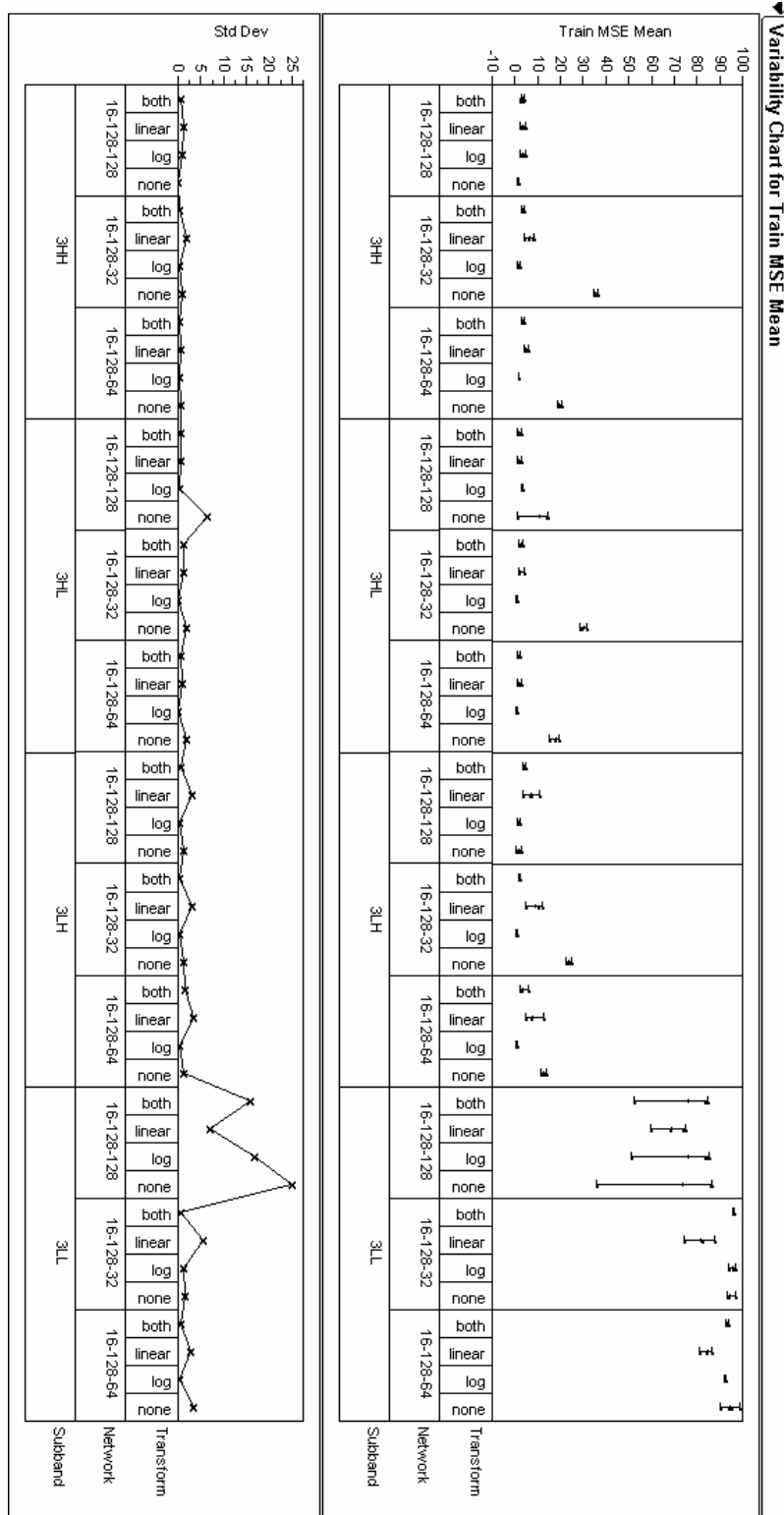


Figure A.3. Level 3 variability chart for the CAD testing set mean square error.

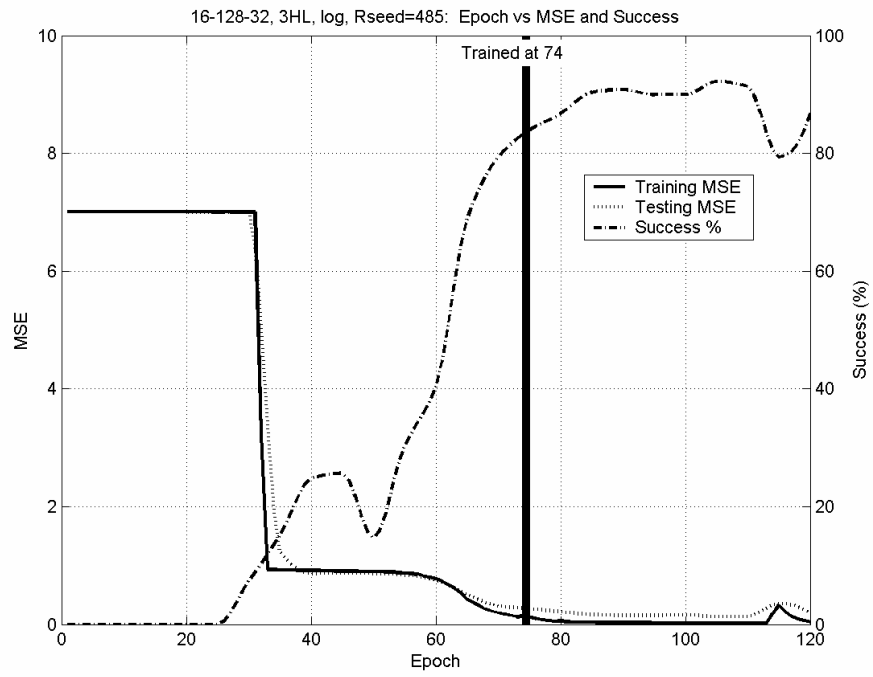


Figure A.4. Level 3, HL subband trained epoch.

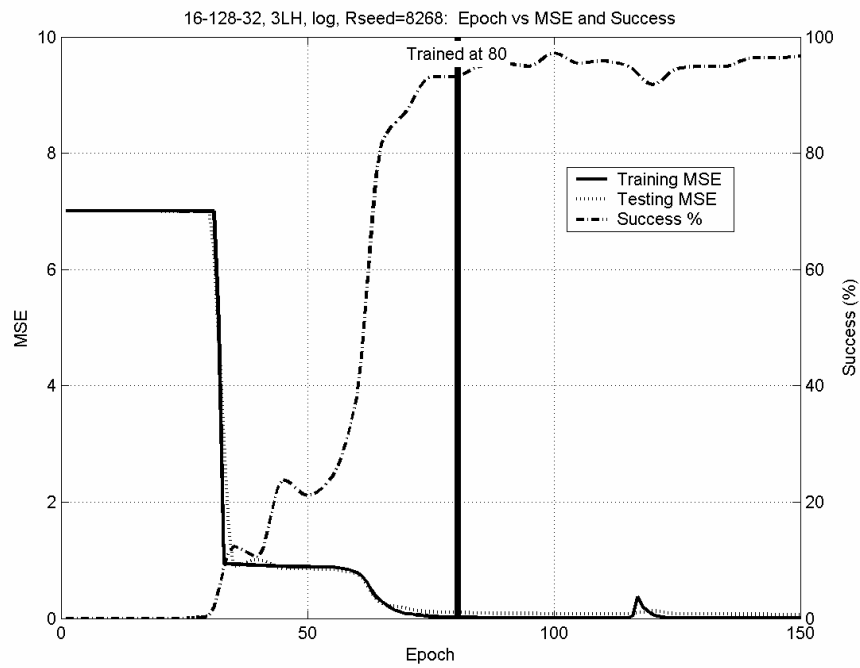


Figure A.5. Level 3, LH subband trained epoch.

## Appendix B: Results for Level 2

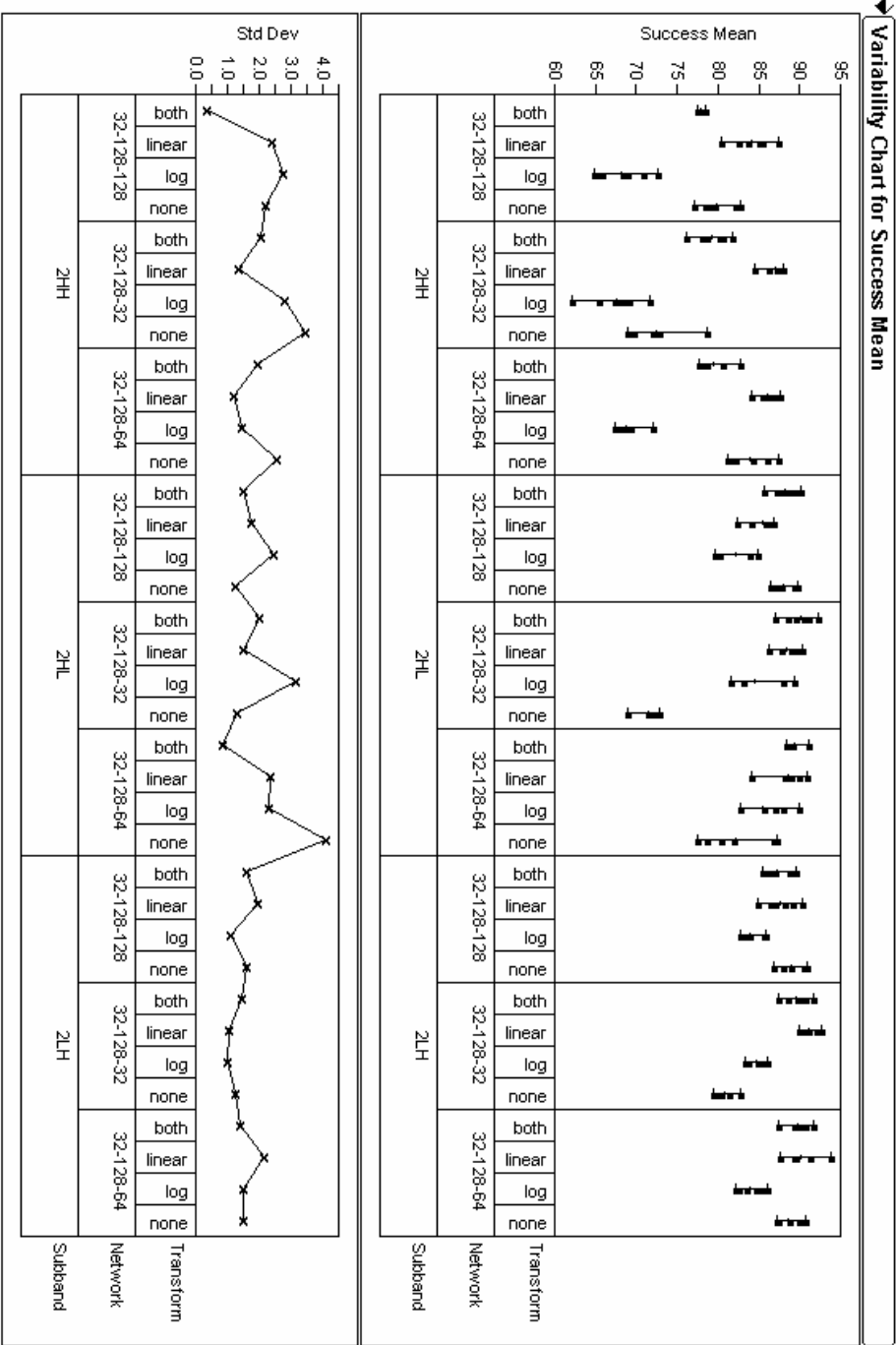


Figure B.1. Level 2 variability chart for the CAD set success rate.

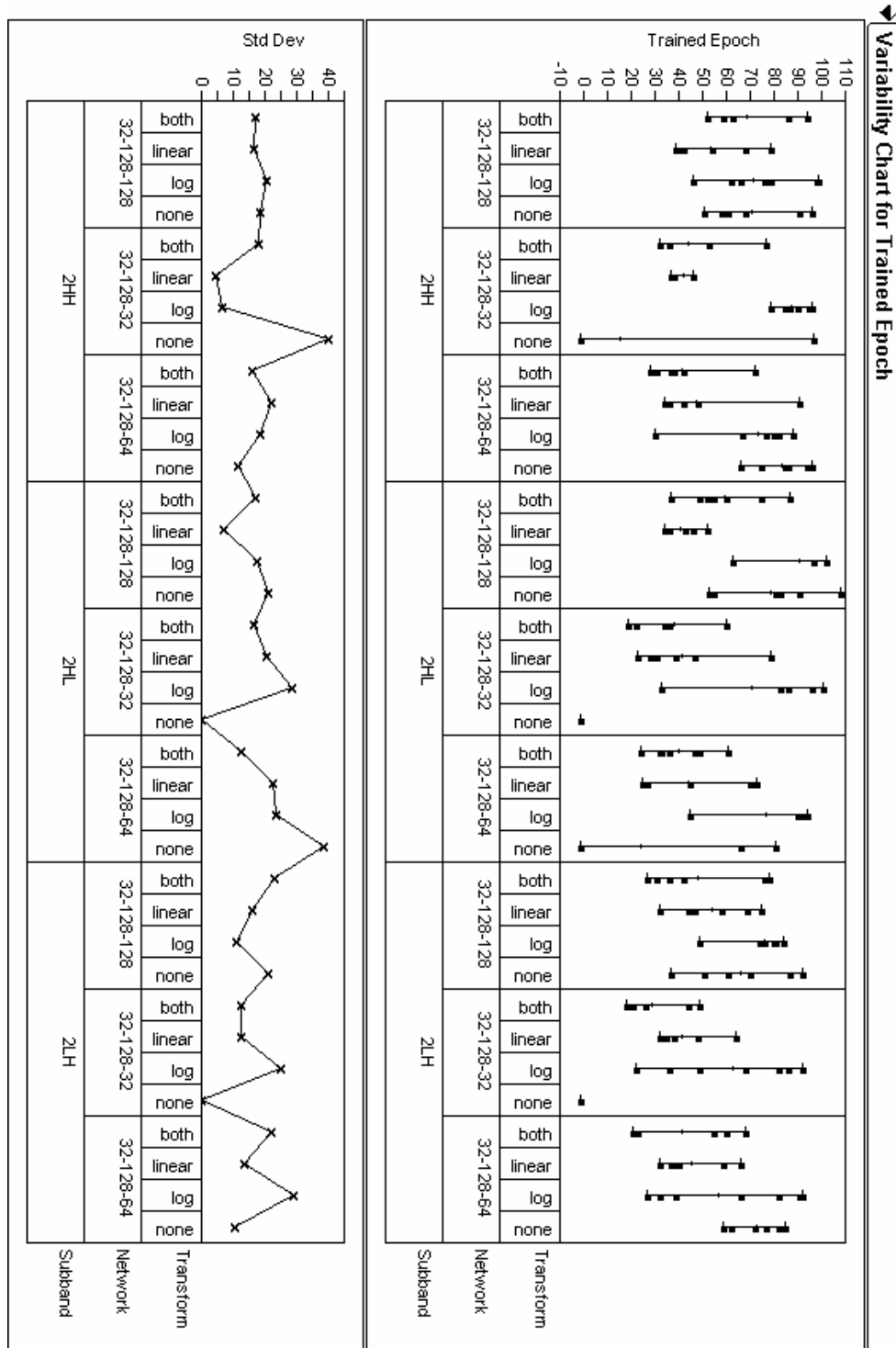


Figure B.2. Level 2 variability chart for the CAD set trained epoch.



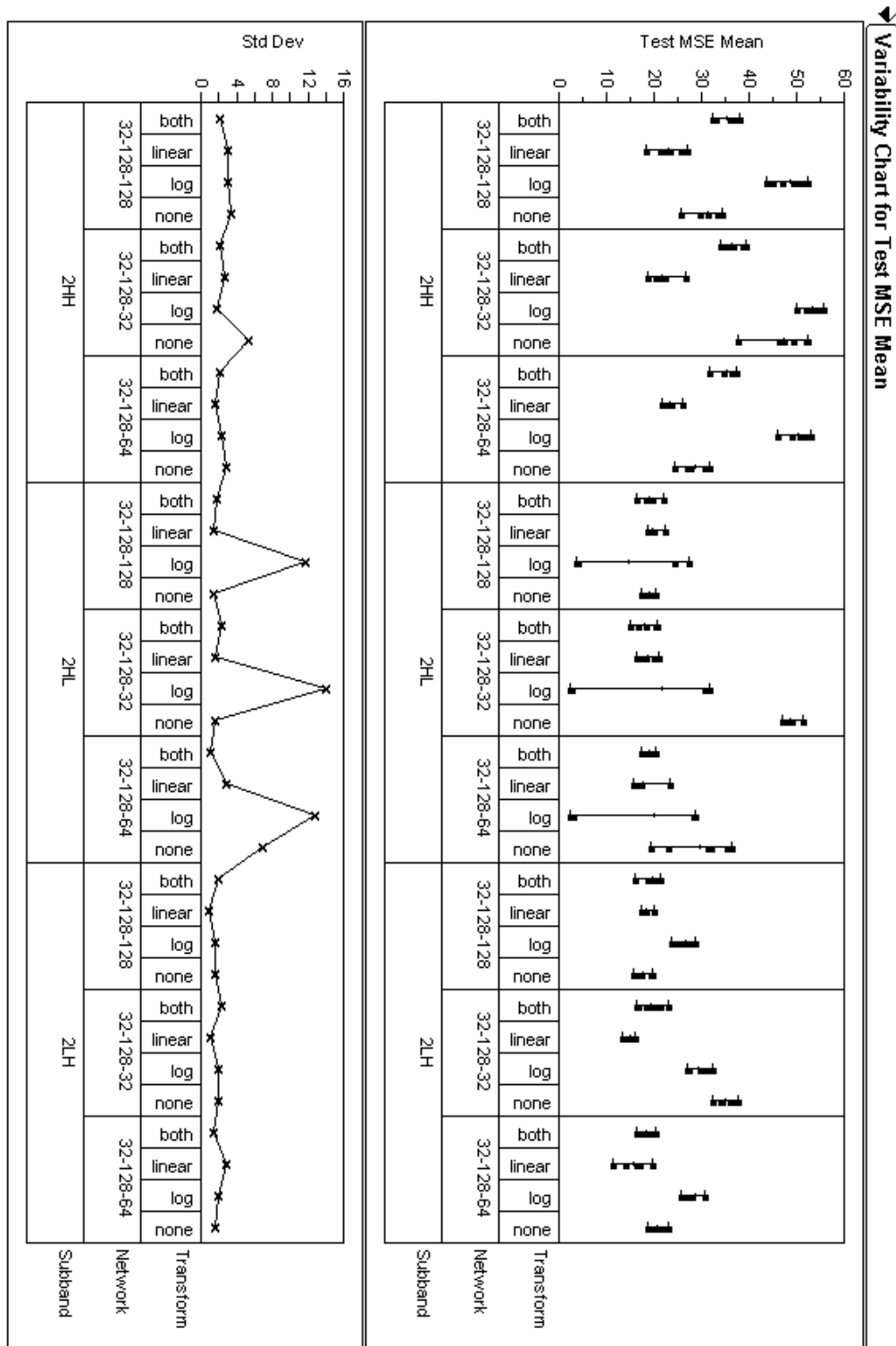


Figure B.3. Level 2 variability chart for the CAD testing set mean square error.

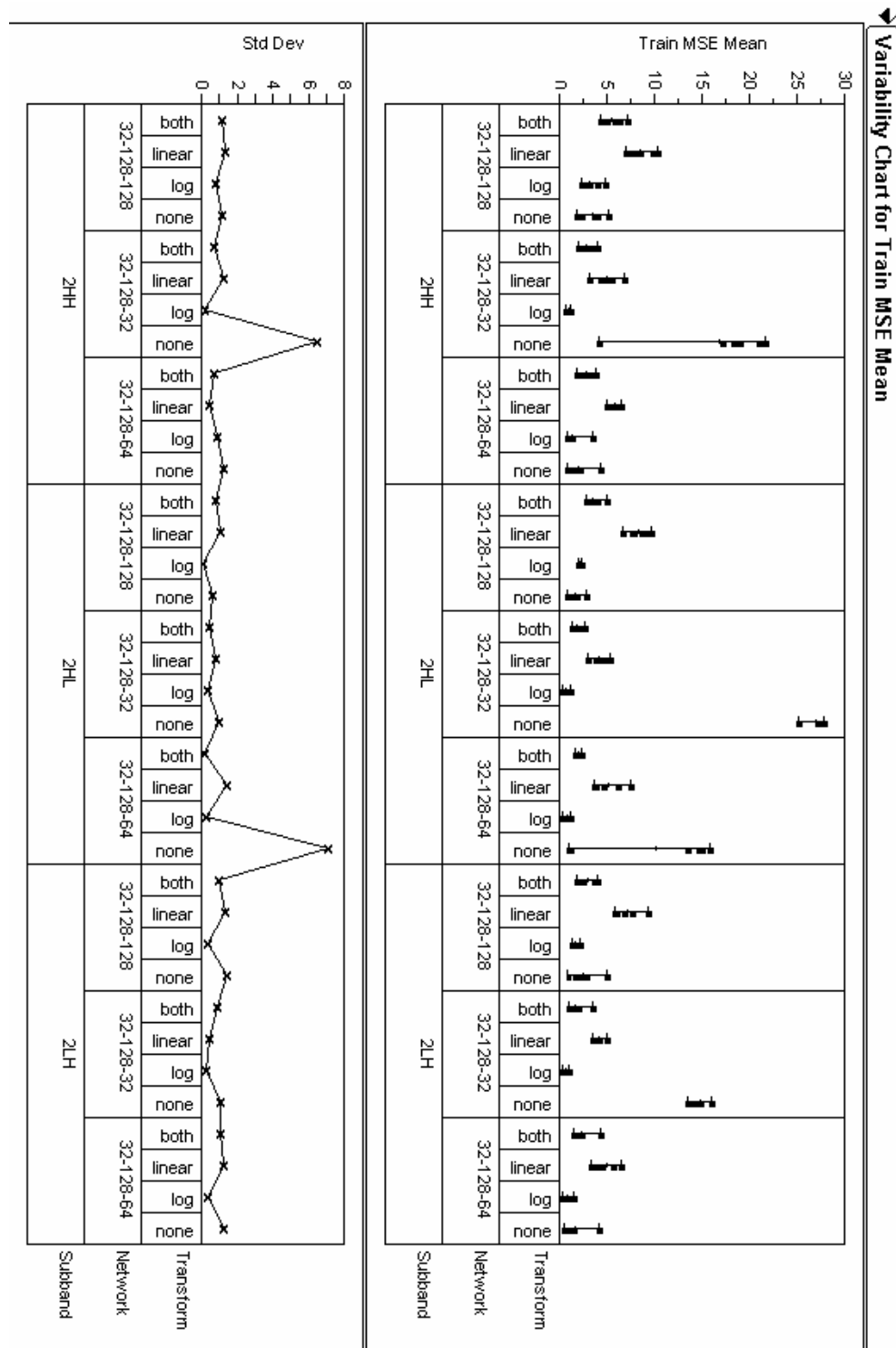


Figure B.4. Level 2 variability chart for the CAD training set for success rate.

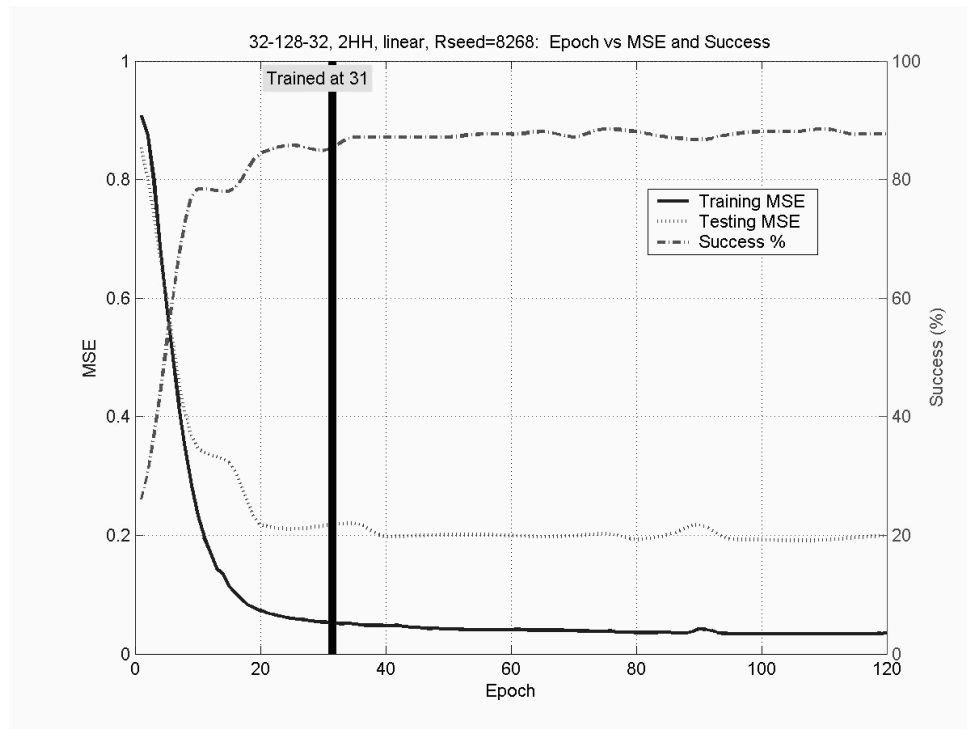


Figure B.5. Level 2, HH subband trained epoch.

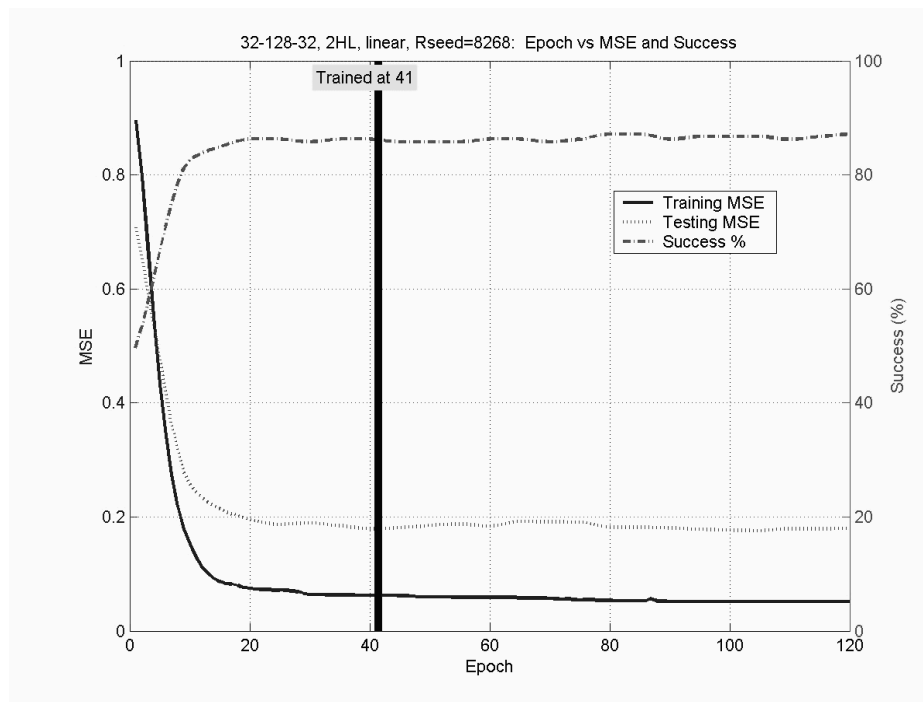


Figure B.6. Level 2, HL subband trained epoch.

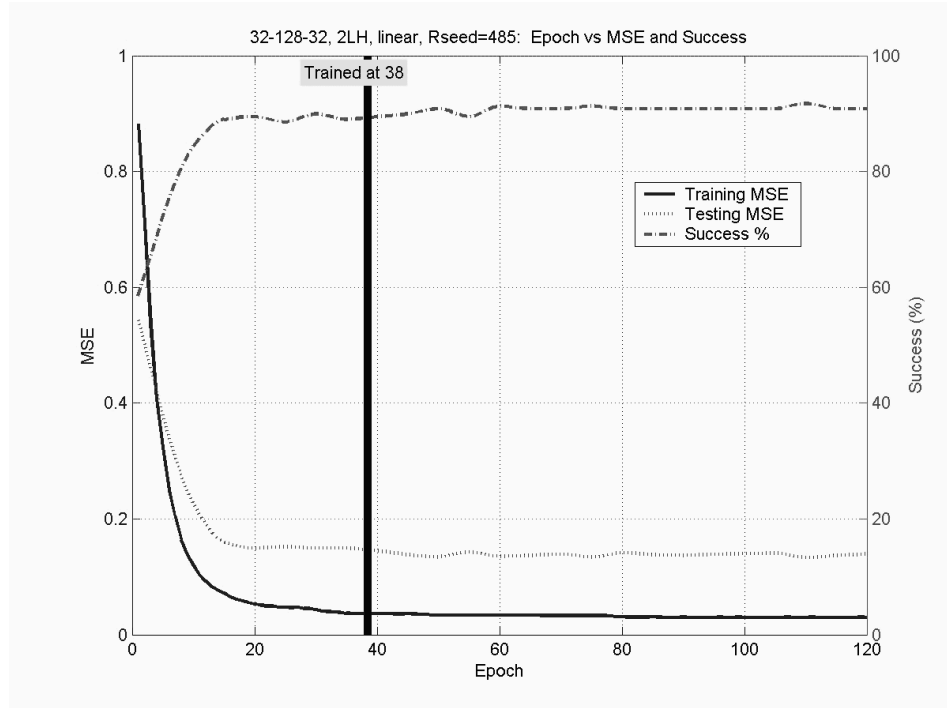


Figure B.7. Level 2, LH subband trained epoch.

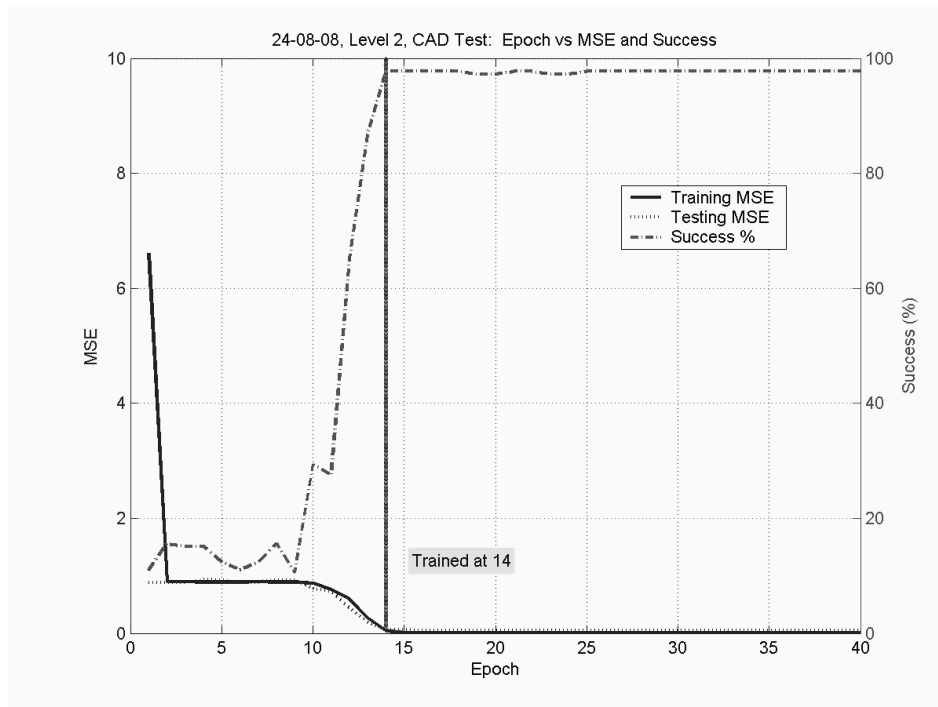


Figure B.8. Results of the CAD testing set on the level 2 “glue” fully connected network with hidden nodes.

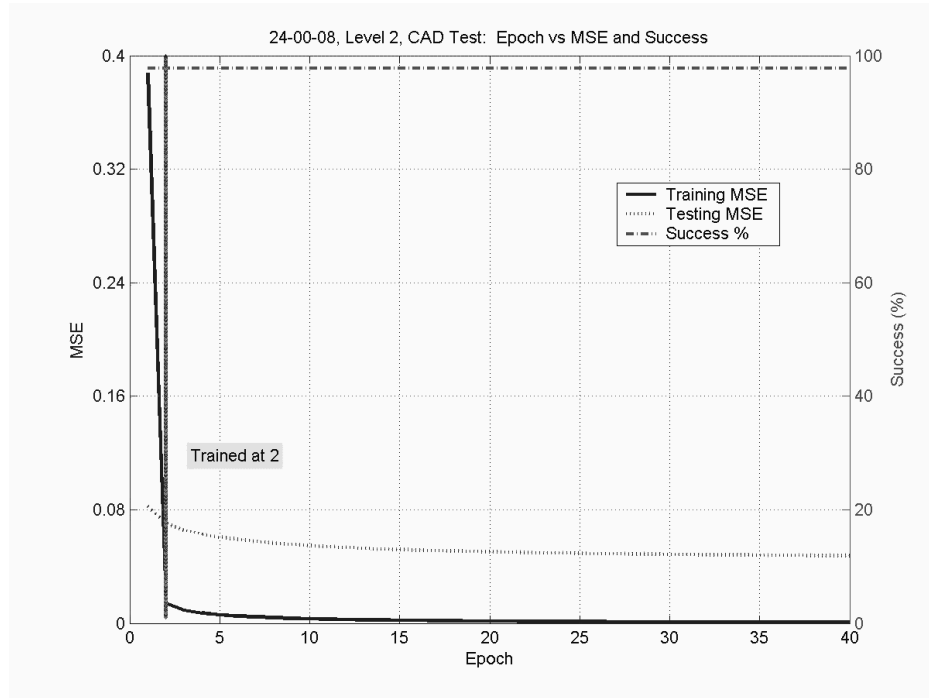


Figure B.9. Results of the CAD testing set on the level 2 “glue” fully connected network without hidden nodes.

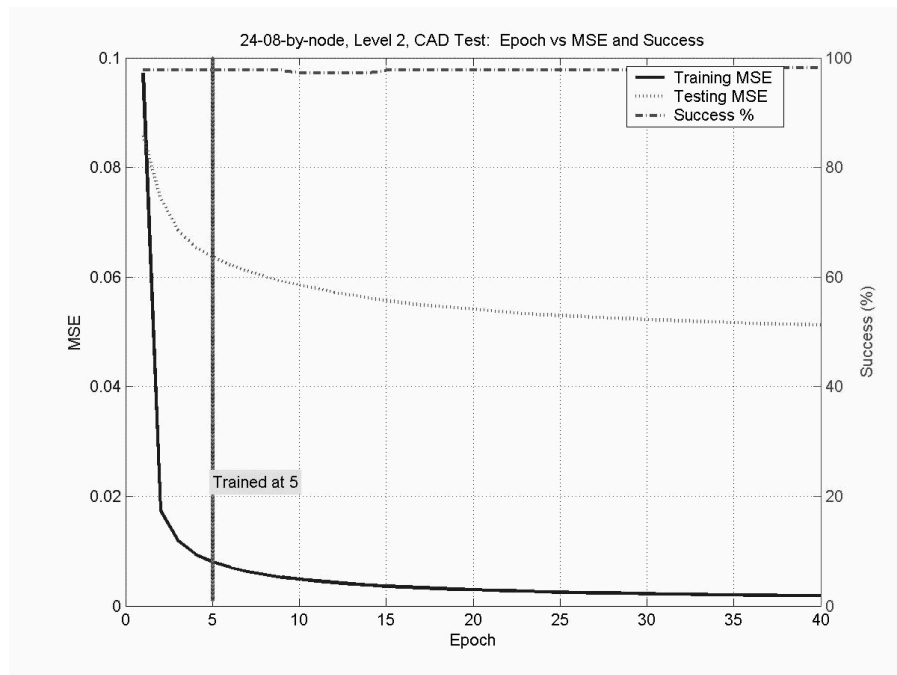


Figure B.10. Results of the CAD testing set on the level 2 correlating network.

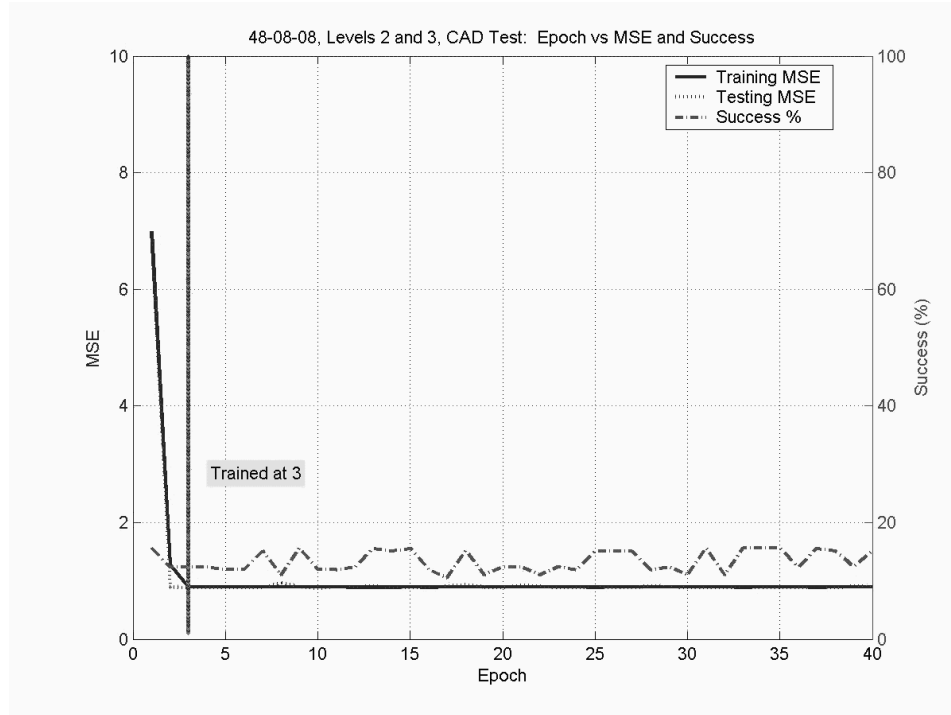


Figure B.11. Results of the CAD testing set on the level 2 and 3 “glue” fully connected network with hidden nodes.

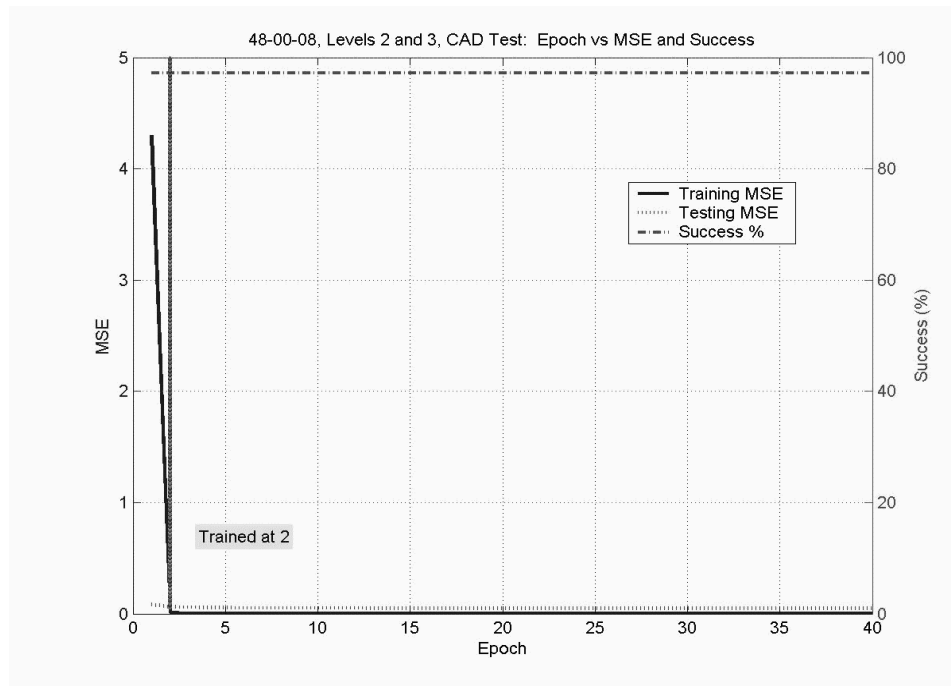


Figure B.12. Results of the CAD testing set on the level 2 and 3 “glue” fully connected network without hidden nodes.

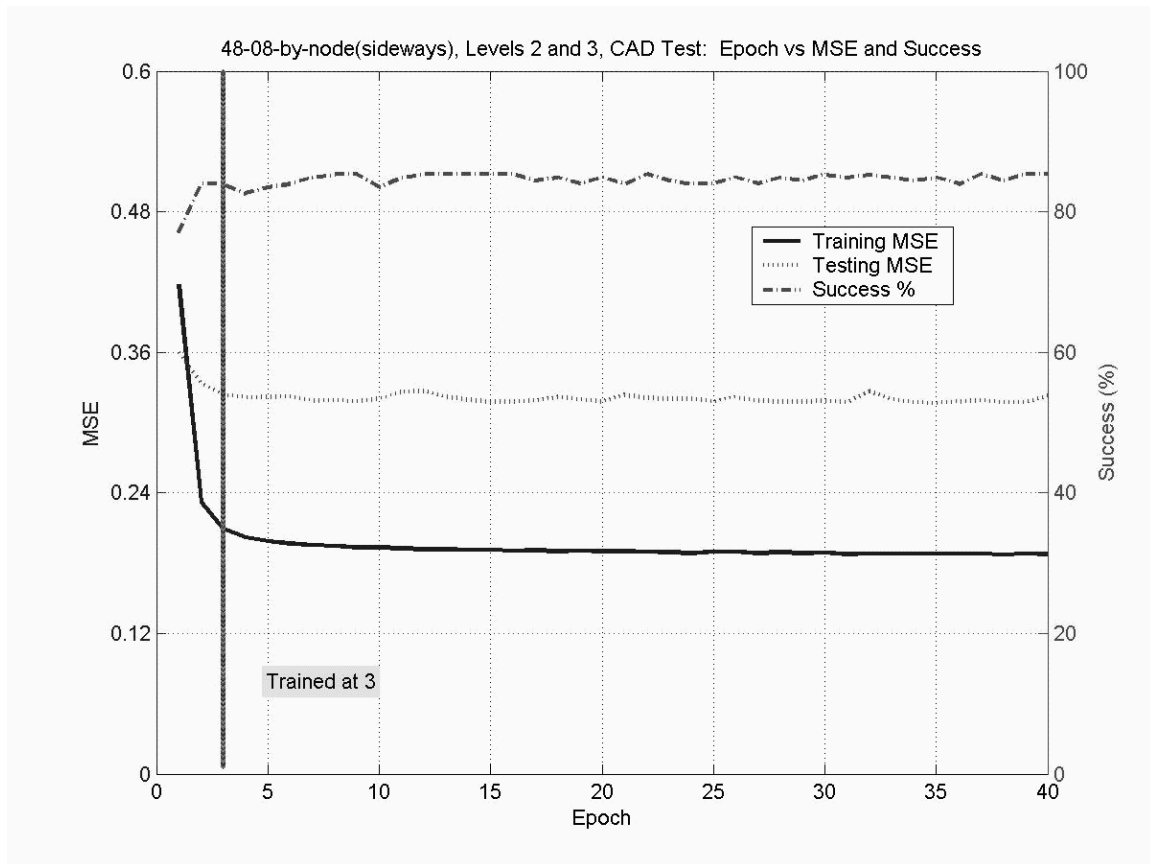


Figure B.13. Results of the CAD testing set on the level 2 and 3 correlating network.

## Appendix C: Results for Level 1

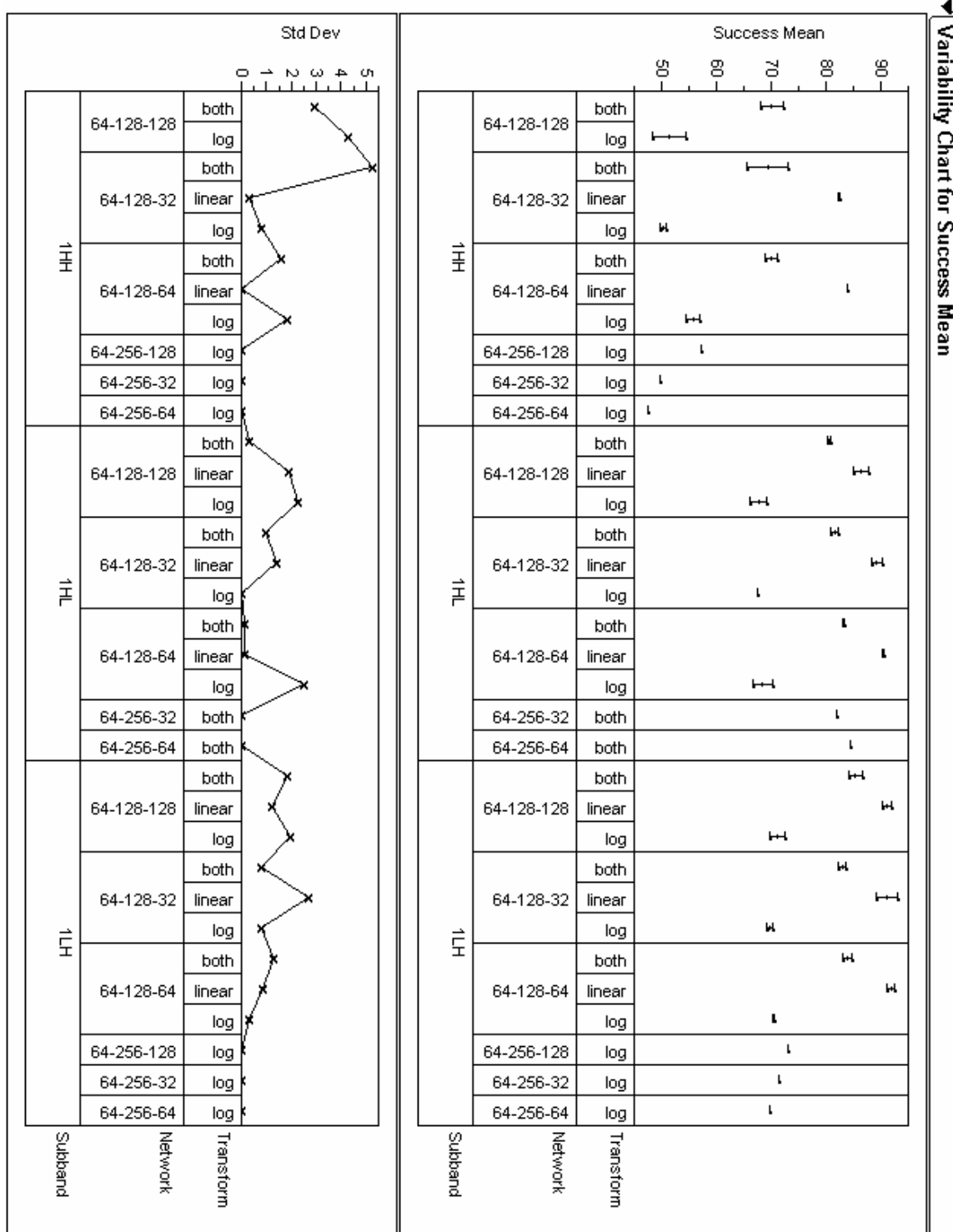


Figure C.1. Level 1 variability chart for the CAD set success rate.



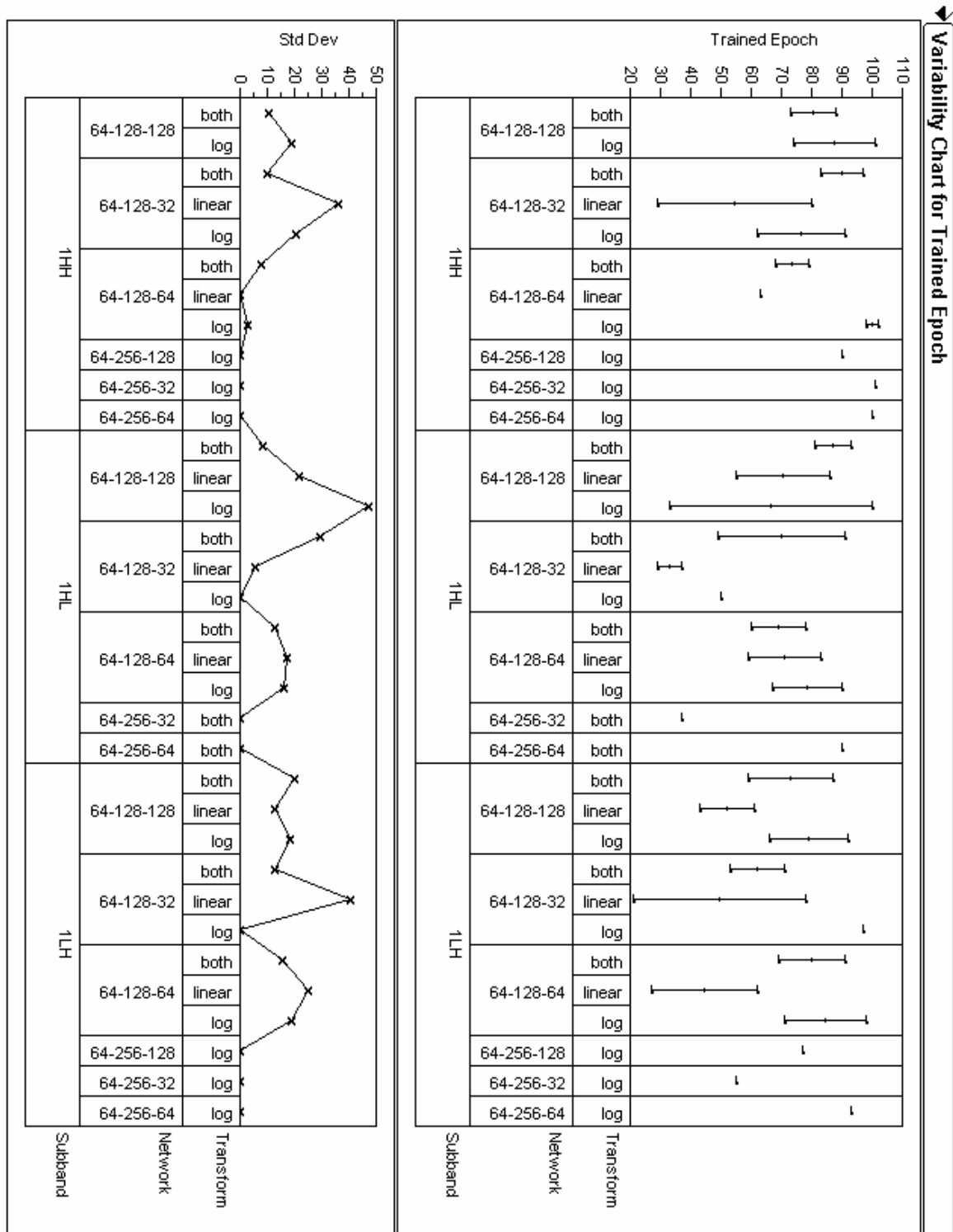


Figure C.2. Level 1 variability chart for the CAD set trained epoch.

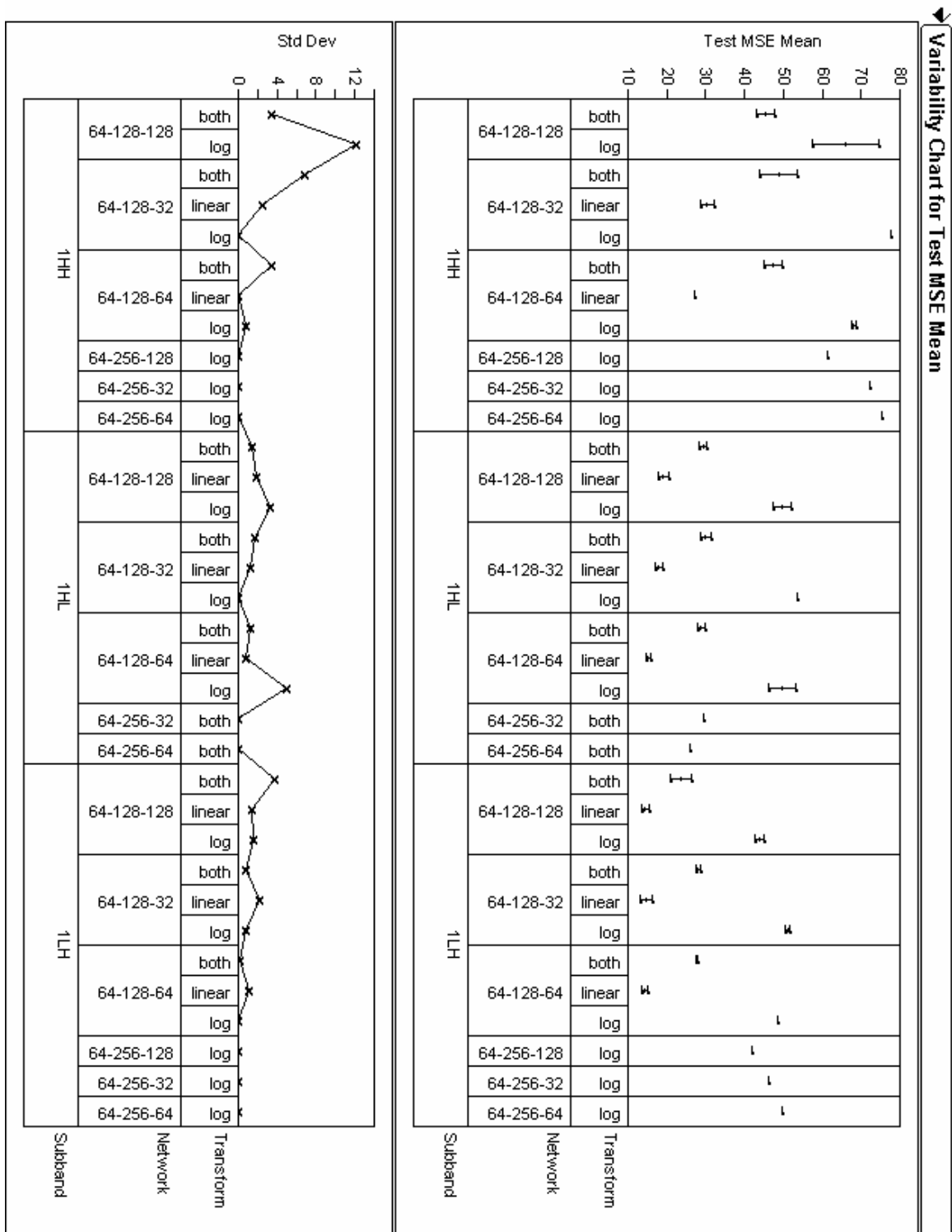


Figure C.3. Level 1 variability chart for the CAD testing set mean square error.

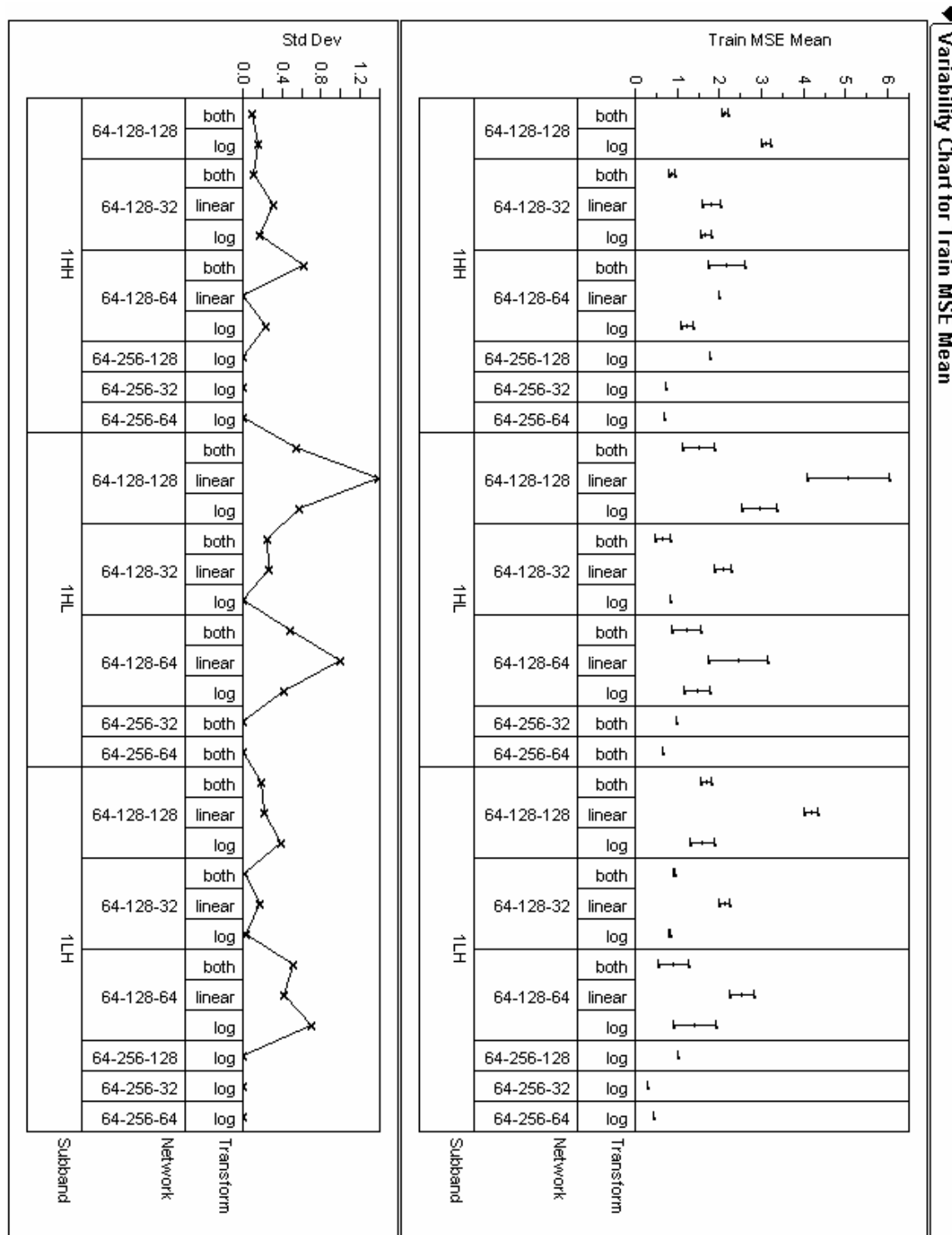


Figure C.4. Level 1 variability chart for the CAD training set mean square error.

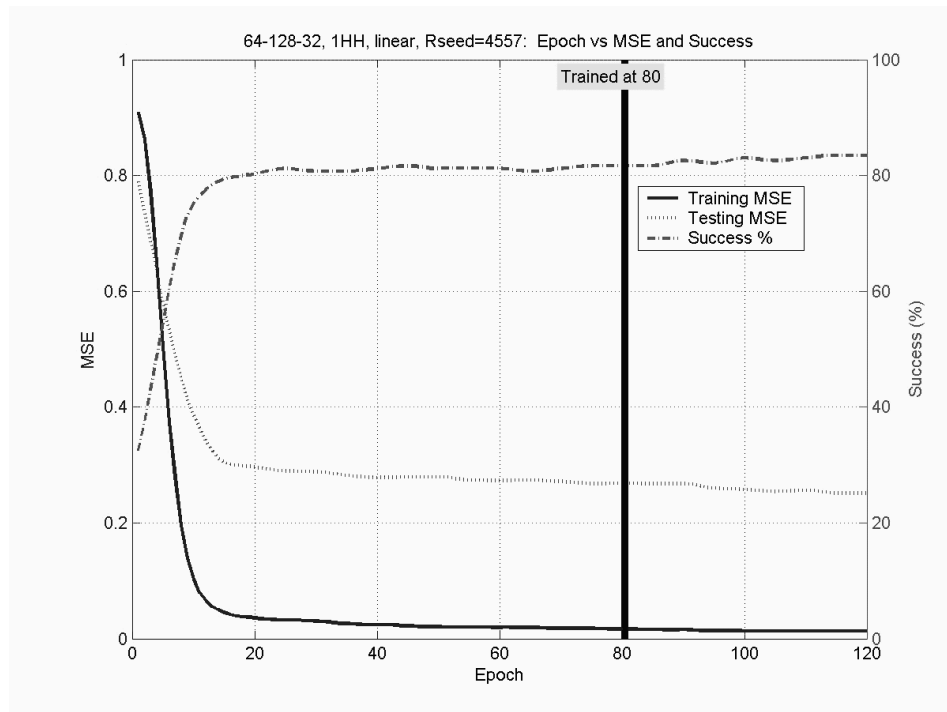


Figure C.5. Level 1, HH subband trained epoch.

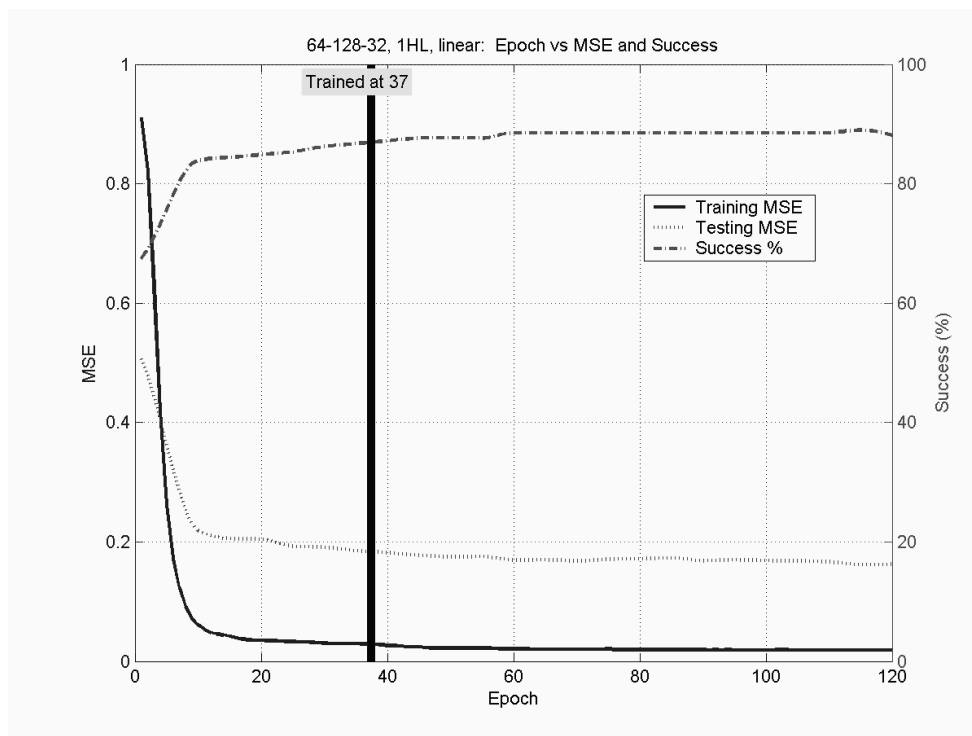


Figure C.6. Level 1, HL subband trained epoch.

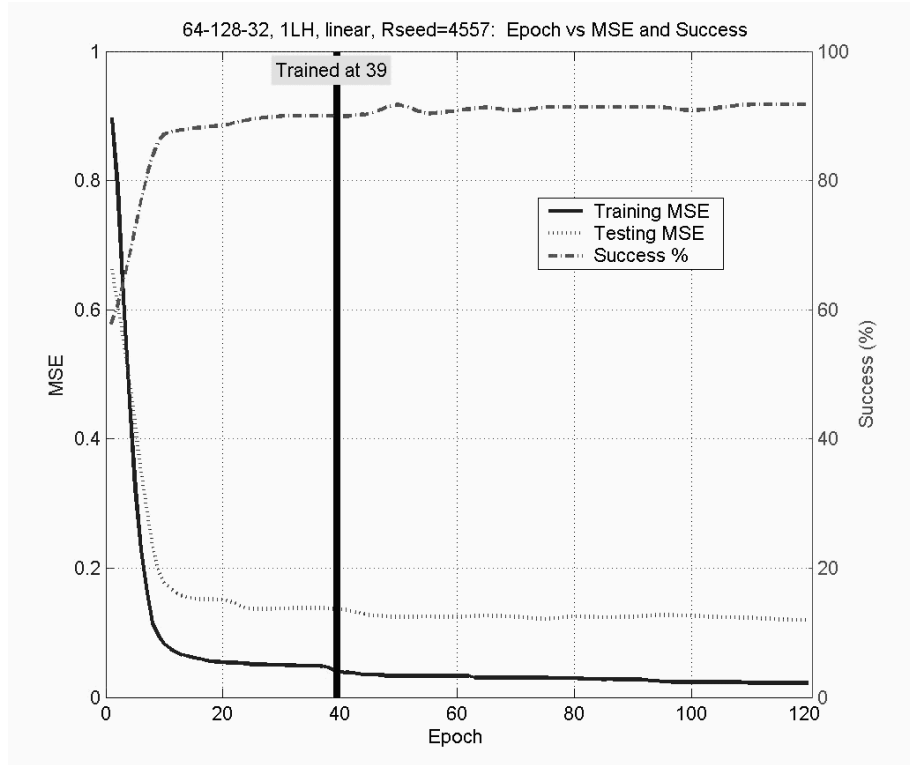


Figure C.7. Level 1, LH subband trained epoch.

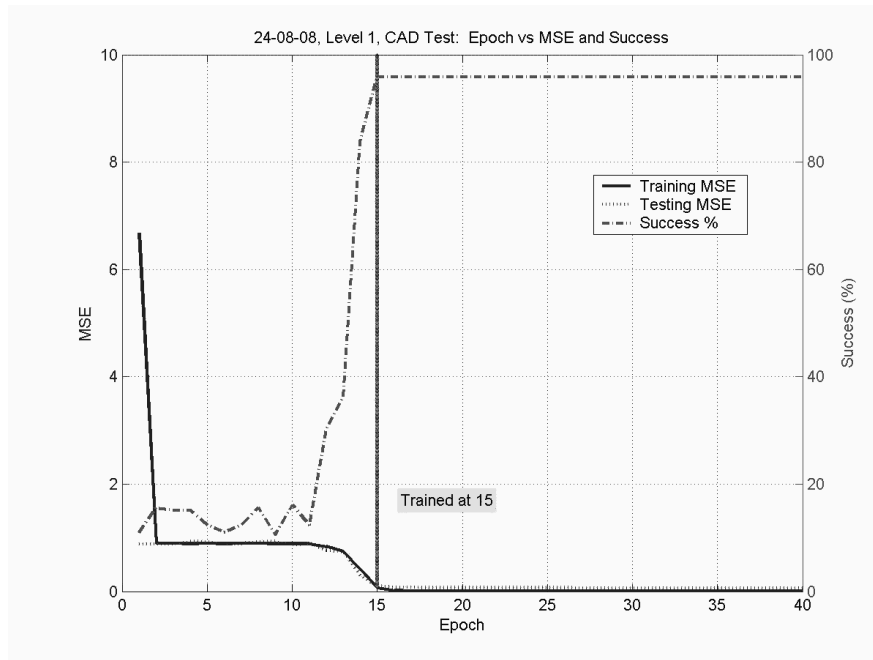


Figure C.8. Results of the CAD testing set on the level 1 “glue” fully connected network with hidden nodes.

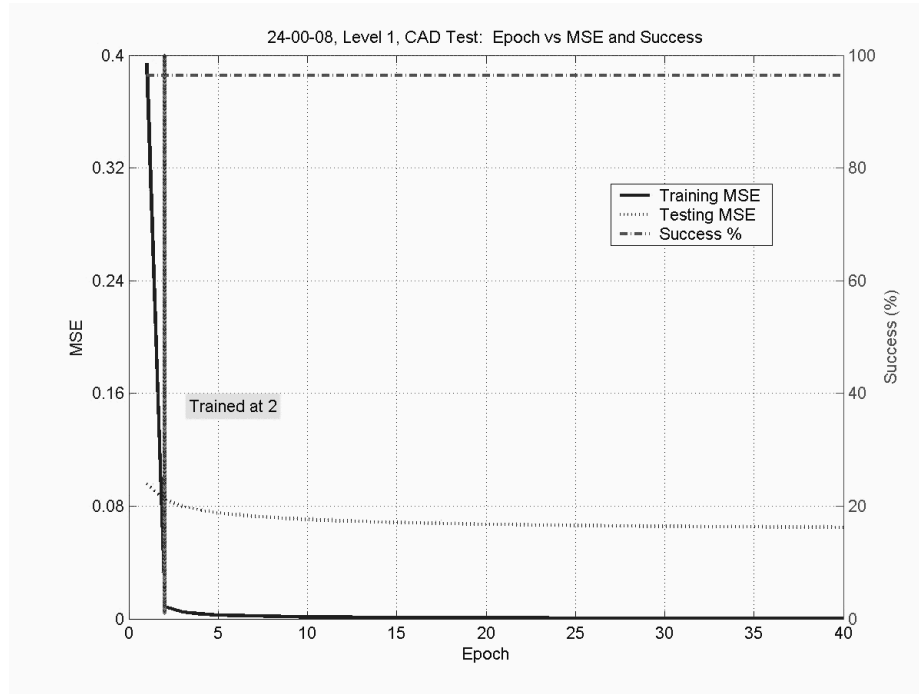


Figure C.9. Results of the CAD testing set on the level 1 “glue” fully connected network without hidden nodes.

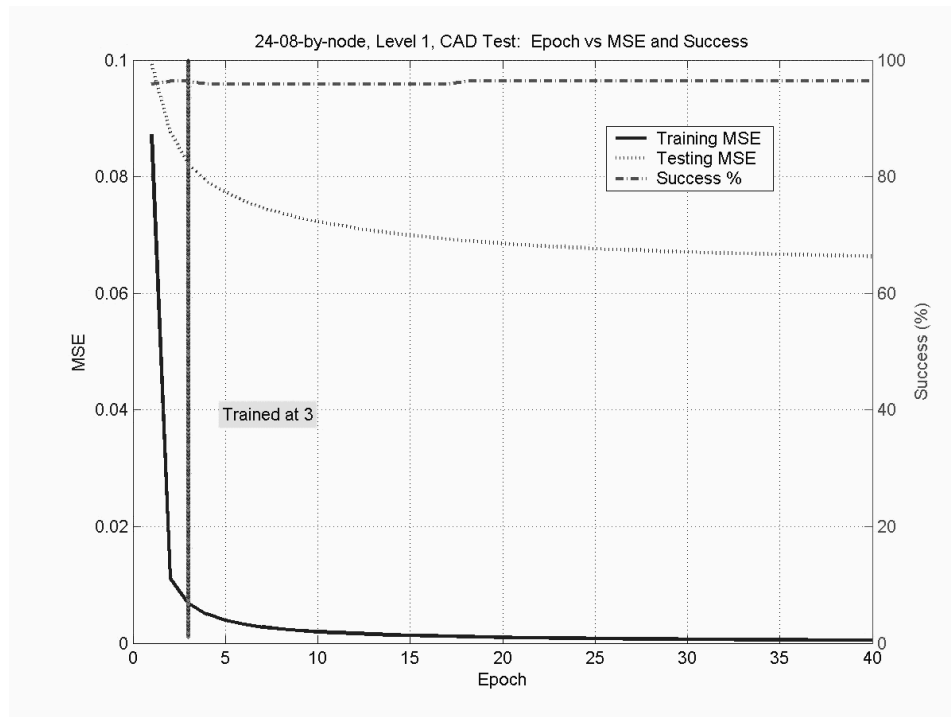


Figure C.10. Results of the CAD testing set on the level 1 correlating network.

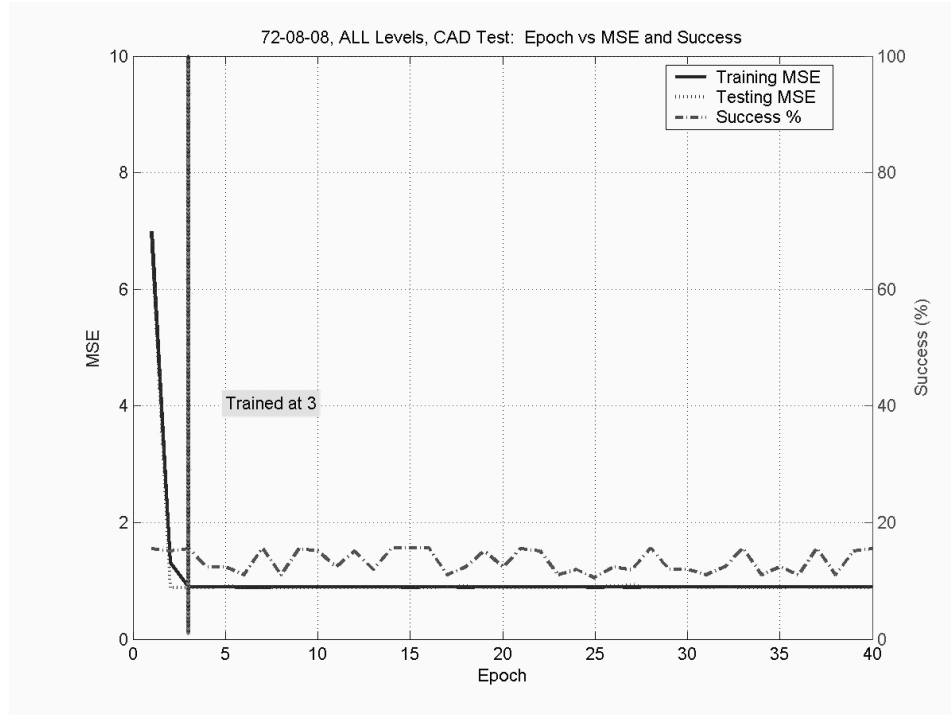


Figure C.11. Results of the CAD testing set on the level 1, 2, and 3 “glue” fully connected network with hidden nodes.

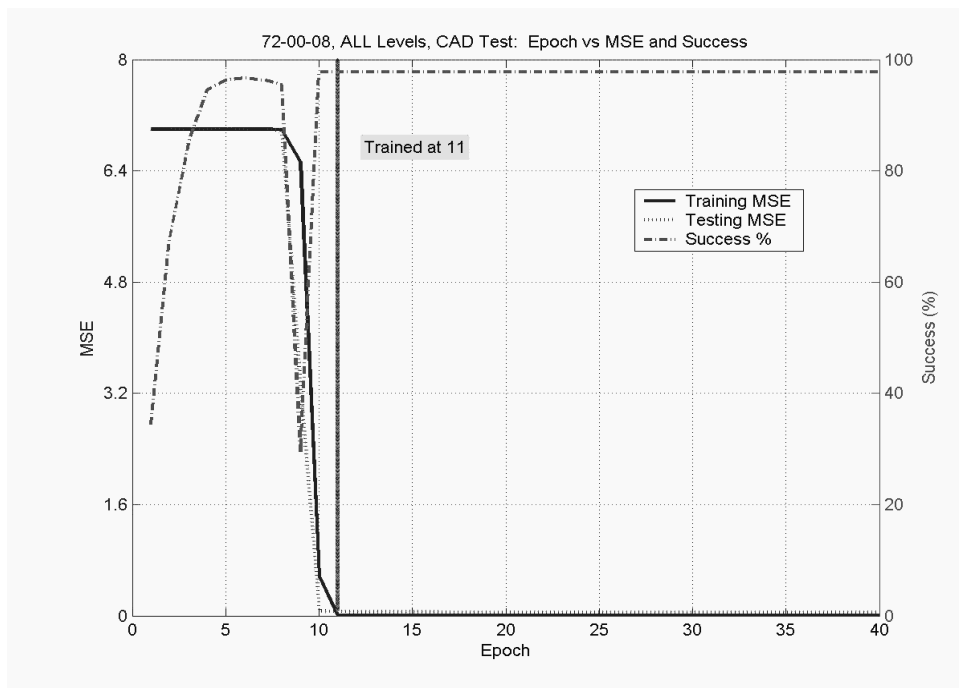


Figure C.12. Results of the CAD testing set on the level 1, 2, and 3 “glue” fully connected network without hidden nodes.

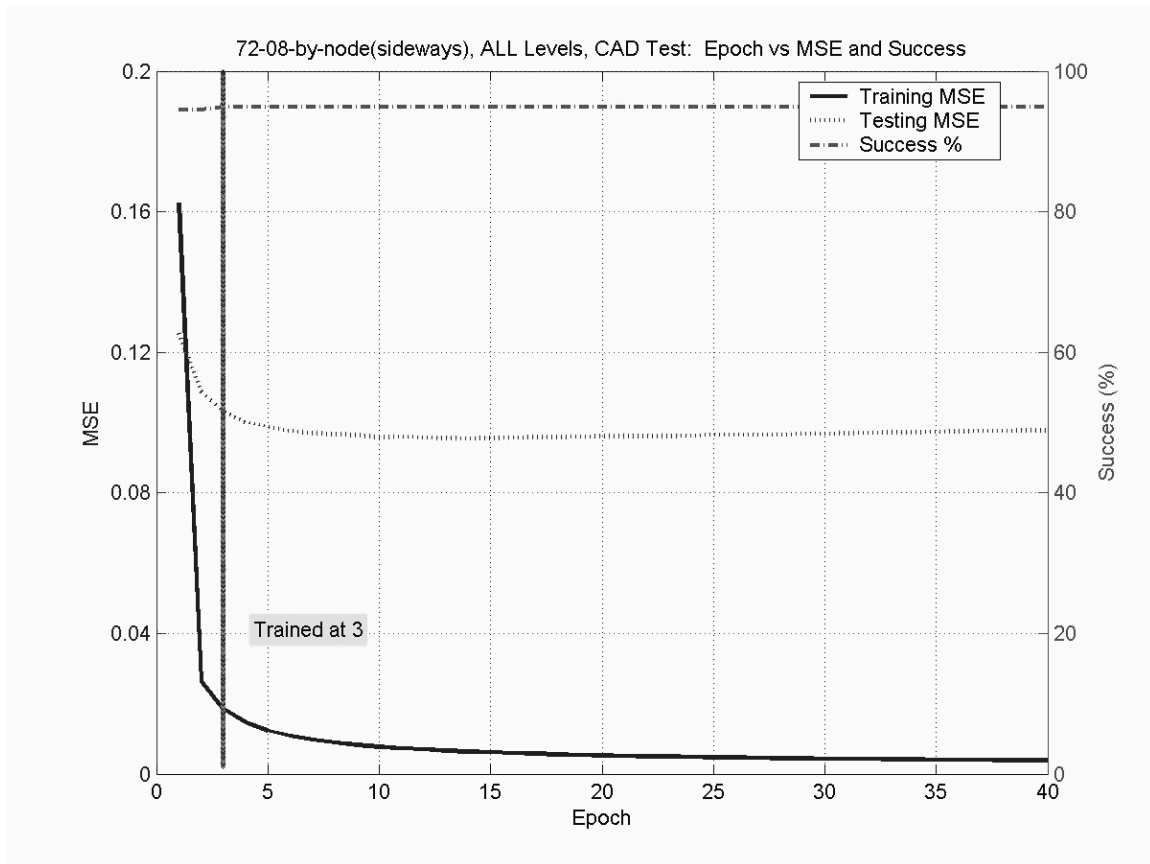


Figure C.13. Results of the CAD testing set on the level 1, 2, and 3 correlating network.



## Appendix D: Results for Training and Testing with Photo Sets

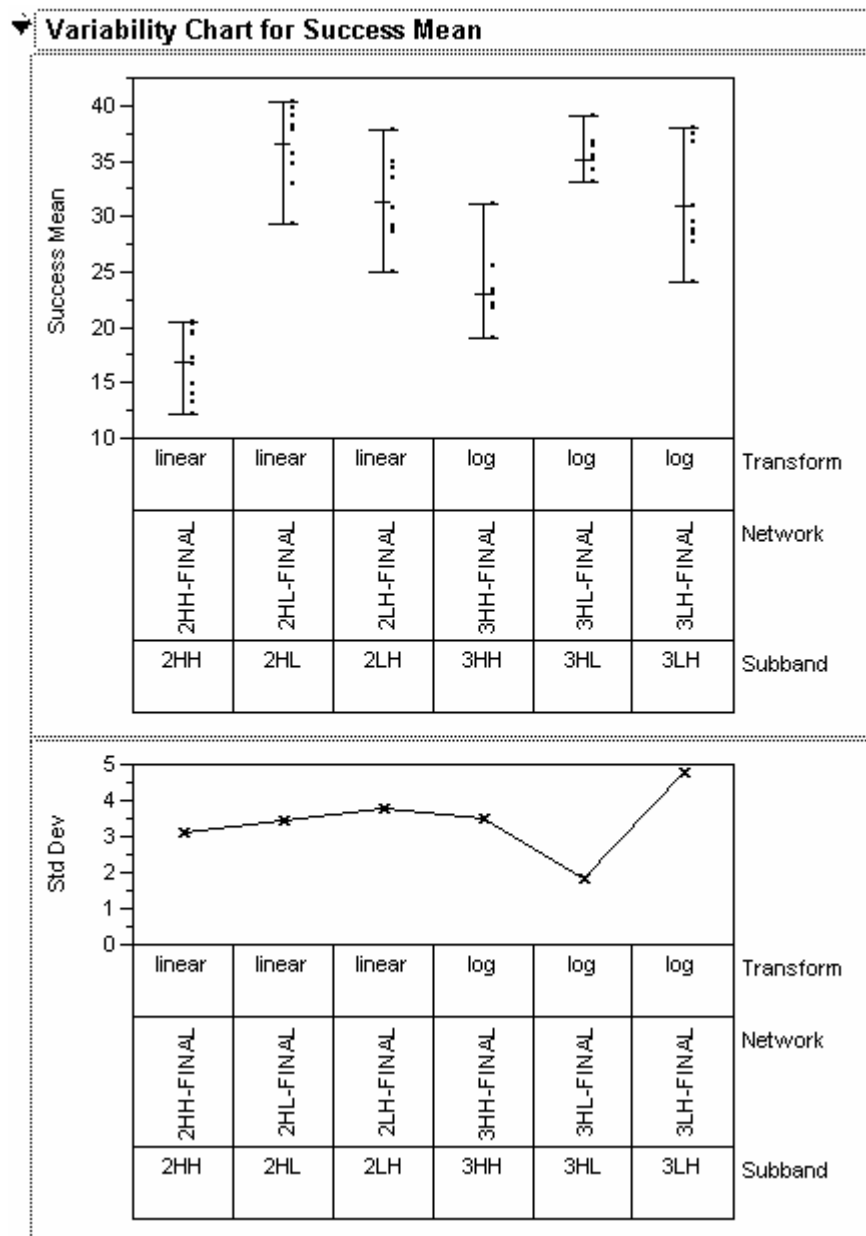


Figure D.1. Variability chart for the photo set success rate.

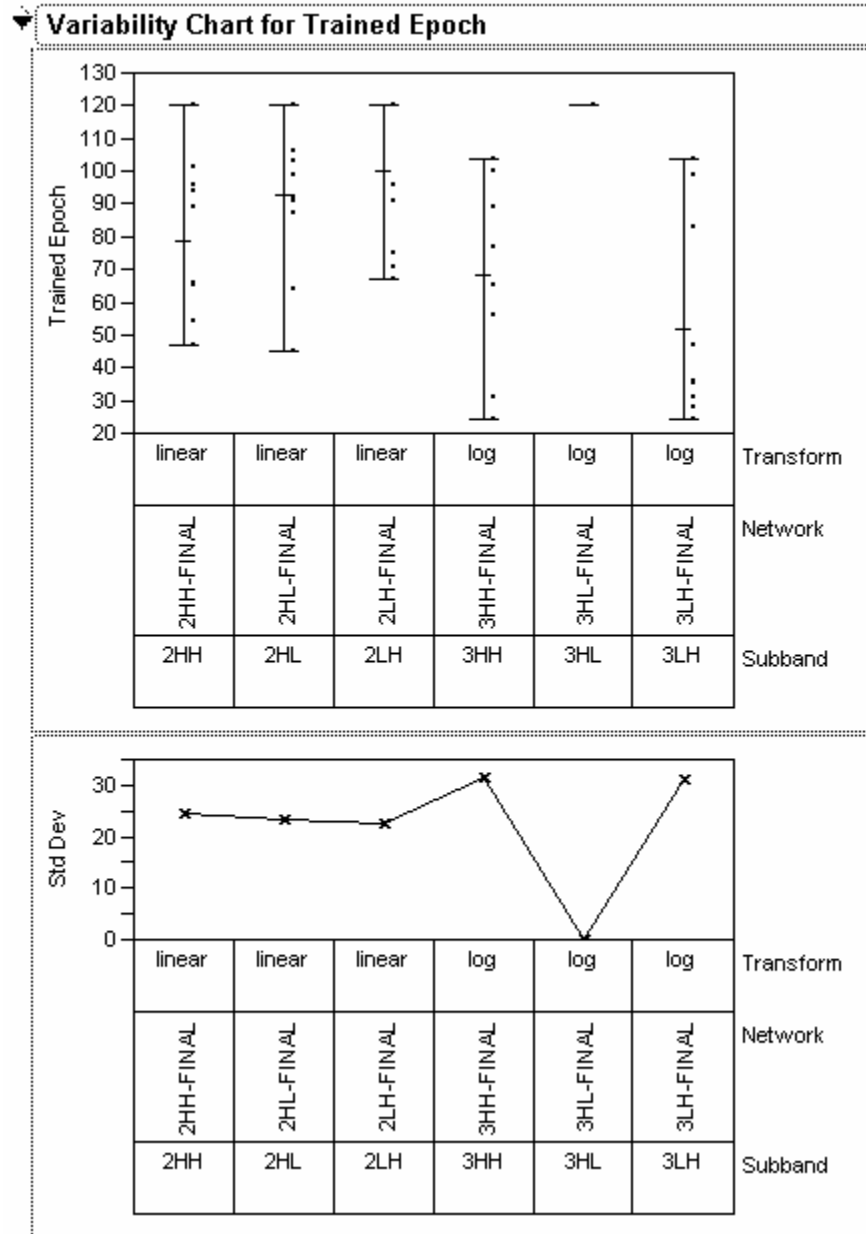


Figure D.2. Variability chart for the photo set trained epoch.

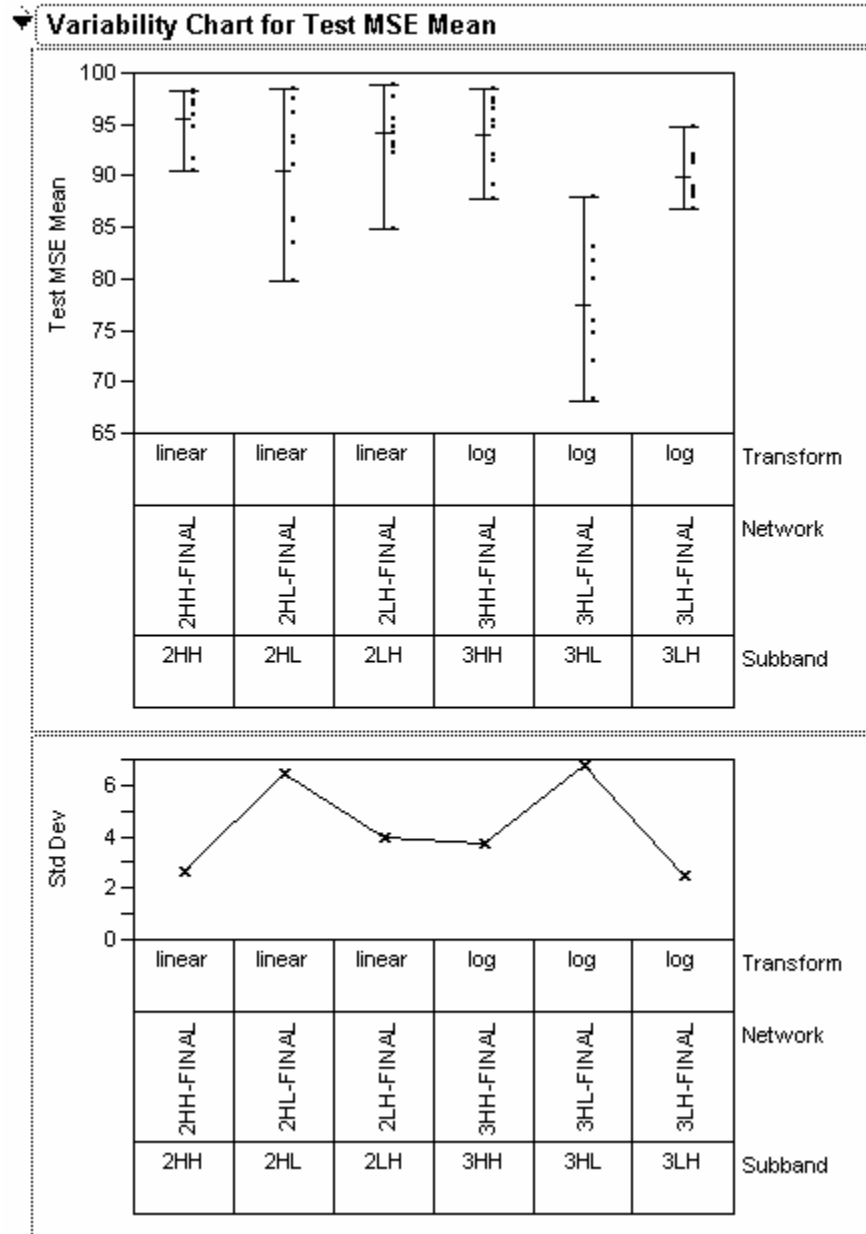


Figure D.3. Variability chart for the photo testing set mean square error.

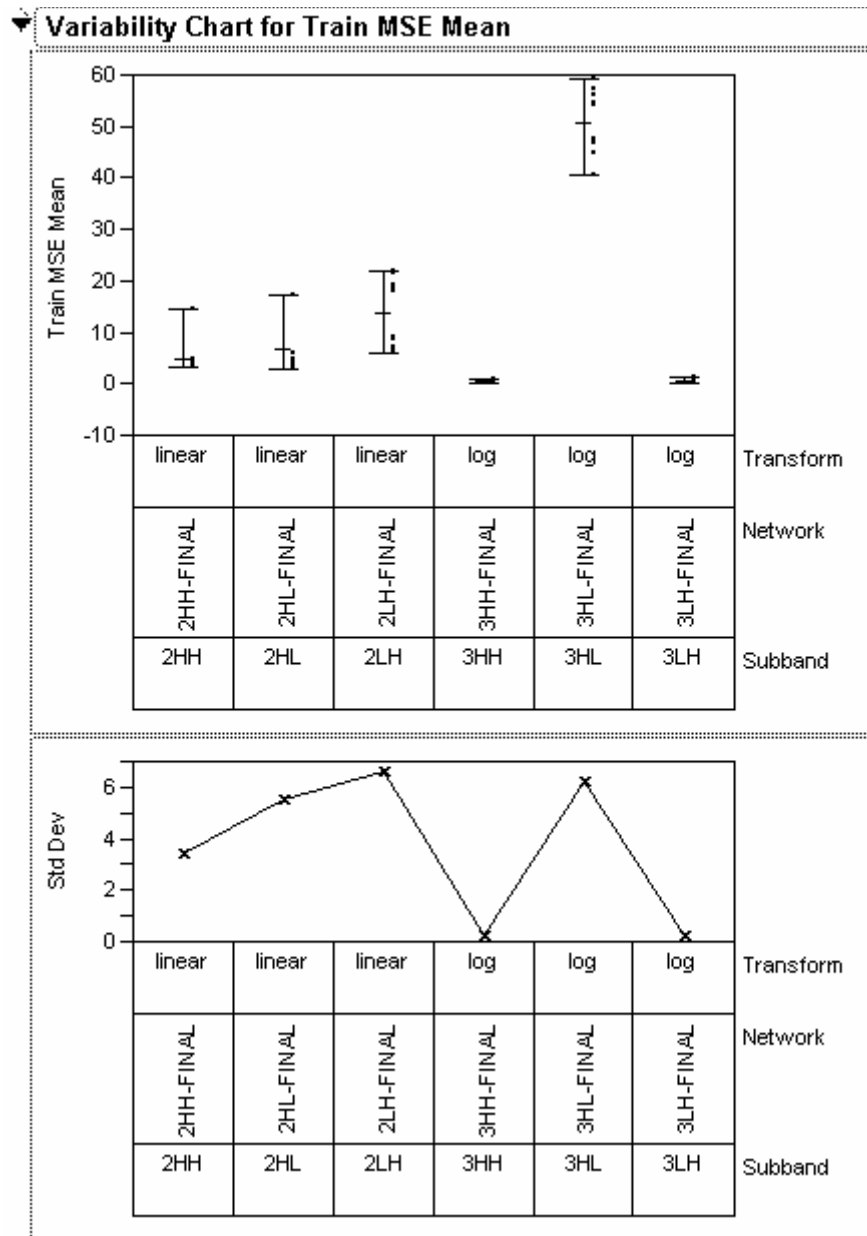


Figure D.4. Variability chart for the photo training set mean square error.

## Appendix E: MATLAB Code for Computing Trained Epoch

This function computes the epoch at which the network is fully trained. The training MSE, test MSE, success rate on the testing set, and a list of the epochs. Since the testing set measurements are only made every five epochs, the values must be interpolated. This algorithm works by computing the first and second derivatives and examining the rates of change. It initially looks for the large initial drop in MSE and then switches to examining the success rate. The network is considered fully trained once the average success rate for the last 10 epochs falls within 1.5% of the remaining values up through epoch 120.

```
function [startidx, vlmean, vlvar, v2mean, v2var, v3mean, v3var] = ...
    choose_cutoff(v1, v2, v3, v4)
% Arguments:
%   v1 = train_mse
%   v2 = test_mse
%   v3 = test_success
%   v4 = train_epoch
% Only v2, v3 should require interpolation.
%
% Returns:
%   startidx = epoch at which the network is fully trained
% The other returned values correspond to the input arguments -
% mean and variance.

WINDOWSIZE = 10; %window size to use when checking success
SUCCESSAVG = .015; %average difference allowed for success (1.5% for
this case)

% This values were determined through multiple tests
DERIV1TEST = -.1; %used for 1st derivative test on MSE
DERIV2TEST = .03; %used for 2nd derivative test on MSE
ENDSUCCESSSIZE = 20; %number of last success samples to use for best
case success

% Sanity check on arguments
if (nargin ~= 4)
    error('Bad argument(s) ')
end

%ensure that the row vectors are changed to column vectors
if (size(v1,1) < size(v1,2) )
```

```

    v1 = flipud(rot90(v1));
end
if (size(v2,1) < size(v2,2) )
    v2 = flipud(rot90(v2));
end
if (size(v3,1) < size(v3,2) )
    v3 = flipud(rot90(v3));
end
if (size(v4,1) < size(v4,2) )
    v4 = flipud(rot90(v4));
end

%scale MSE up so that it is easier to test (and appears better if
plotted)
scalefac = 100 / max(max(v1),max(v2));
v1 = v1 * scalefac;
v2 = v2 * scalefac;

%make all vector the same length using cubic interpolation
v2 = interp1([1: size(v4,1)/size(v2,1) : size(v4,1)], v2, v4, 'cubic');
v3 = interp1([1: size(v4,1)/size(v3,1) : size(v4,1)], v3, v4, 'cubic');

%calculate 1st and 2nd derivatives for each, must do fit to create
%an expression that can be differentiated
modelv1 = fit(v4,v1,'cubicspline');
modelv2 = fit(v4,v2,'cubicspline');
%modelv3 = fit(v4,v3,'cubicspline');    %not used

[der1_v1,der2_v1] = differentiate(modelv1,v4);
[der1_v2,der2_v2] = differentiate(modelv2,v4);
%[der1_v3,der2_v3] = differentiate(modelv3,v4);    %not used

%Get past big initial drop, analyze derivatives to find end of major slope
for startidx=1:size(v4,1)
    %fprintf('Deriv (%d)    TrainMSE: %8.3f, %8.3f    Test MSE: %8.3f,
%8.3f\n',startidx, der1_v1(x),der2_v1(x),der1_v2(x),der2_v2(x) )
    if ( (der1_v1(startidx) >= DERIV1TEST) && (abs(der2_v1(startidx)) <=
DERIV2TEST) && ...
        (der1_v2(startidx) >= DERIV1TEST) && (abs(der2_v2(startidx)) <=
DERIV2TEST) )
        break
    end
end

%Ensure that the starting place for next loop reasonable
if (startidx < WINDOWSIZE), startidx = WINDOWSIZE+1; end

if (startidx < (size(v4,1)-WINDOWSIZE) )
    %Find where success levels out - tests the average for the 'window'
and compares
    %it to the average of the last ENDSUCCESSSIZE samples. If within
SUCCESSAVG percent

    %then consider stop training

```

```

    for startidx2=startidx: (size(v4,1)-WINDOWSIZE)
        winavg = mean(v3(startidx2:startidx2+WINDOWSIZE));
        endavg = mean(v3(size(v3,1)-ENDSUCCESSSIZE:size(v3,1) ) );
        fprintf('Avg at (%d)    Success: %8.3f, %8.3f    Diff =
%5.2f%%\n', ...
            startidx2, winavg, endavg, abs(1-winavg/endavg)*100 )
        if (( 1-winavg/endavg) < SUCCESSAVG )
            break
        end
    end
    startidx = max(startidx,startidx2);
    if (startidx >= (size(v4,1)-WINDOWSIZE) )
        startidx = -1; %error code for "not trained"
    end
end

%statistics for the samples starting at cutoff and going to end
if (startidx >= (size(v4,1)-WINDOWSIZE) ) %not trained so get stats
over all epochs
    startidx = 1;
end
endidx = size(v4,1);
v1var = var( v1( startidx : endidx ) );
v1mean = mean(v1( startidx : endidx ) );
v2var = var( v2( startidx : endidx ) );
v2mean = mean(v2( startidx : endidx ) );
v3var = var( v3( startidx : endidx ) );
v3mean = mean(v3( startidx : endidx ) );

%print results to screen
fprintf('
          Train MSE          | Test MSE          |
Success\n')
fprintf('
          Mean          Variance | Mean          Variance | Mean
Variance  \n')
fprintf('(%3d-%3d) %6.4f      %8.5f |%6.4f      %8.5f | %6.3f
%8.3f\n\n', ...
        startidx, endidx, v1mean, v1var, v2mean, v2var, v3mean, v3var)

if (startidx == 1)
    startidx = -1;
end

return

```

## Bibliography

- [Ahme74] Ahmed, N., T. Natarajan, and K. Rao. "Discrete Cosine Transforms," *IEEE Transaction on Computers*, C-23:90-93 (Jan. 1974).
- [Alsp92] Alspector, J., A. Jayakumar, and S. Luna. "Experimental Evaluation of Learning in a Neural Microsystem," *Advances in Neural Info. Proc. Sys.* 4:871-878 (1992).
- [Burr98] Burrus, C. and others. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Upper Saddle River, NJ: Prentice Hall, 1998.
- [Clay99] Claypool, R. "Daubechies (7,9) Implementation using the Wavelet Lifting Scheme." MATLAB Script, 1999.
- [Ches90] Chester, D. "Why Two Hidden Layers are Better Than One," *International Joint Conference on Neural Networks*, 1:265-268. Washington, DC, 1990.
- [Corr02] Correia, S., J. Carvalho, and R. Sabourin. "On the Performance of Wavelets for Handwritten Numerals Recognition," *ICPR*, (2002).
- [Debn02] Debnath, L. *Wavelet Transforms & Their Applications*. Boston, Basel, Berlin: Birkhauser, 2002.
- [Faug93] Faugeras, O. *Three-Dimensional Computer Vision: A Geometric Viewpoint*," MIT Press: Cambridge, Massachusetts (1993).
- [Fell91] Felleman, D. and D. van Essen. "Distributed Hierarchical Processing in the Primate Cerebral Cortex," *Cerebral Cortex*, 1:1-47 (1991).
- [Funa89] Funahashi, K. "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, 2:183-192 (1989).
- [Hayk94] Haykin, S. *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing Company, Inc., 1994.
- [Hert91] Hertz J, A. Krogh, and Palmer R. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA., 1991.
- [Hopc79] Hopcroft, J. and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Reading, Mass.: Addison-Wesley, 1979.
- [Irfa03] Skiljan, I. "IrfanView." Version 3.85, Windows. Computer Software. Available at <http://www.irfanview.com>, Accessed on 18 Jan 2003.



- [Jack03] Jackson, J. *Targeting Covert Messages: A Unique Approach for Detecting Novel Steganography*. MS thesis, AFIT/GCE/ENG/03-02, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 2003.
- [Jeli99] Jelinek, D. and C. Taylor. "Reconstructing Linearly Parameterized Models from Single View with a Camera of Unknown Focal Length," *Proceedings of Int. Conference on Computer Vision & Pattern Recognition*, 1999: 346-352.
- [Karr97] Karras, D., S. Karkanis, and B. Mertzios. "Information Systems Based on Neural Network and Wavelet Methods with Application to Decision Making, Modelling, and Prediction Tasks," *23<sup>rd</sup> Euromicro Conference*. Budapest, Hungary. 1997.
- [Kast91] [edited by] Kasturi, R. and R. Jain. *Computer Vision: Principles*, IEEE Computer Society Press Tutorial, Los Alamitos, California (1991).
- [Kros96] Krose, B. and P. van der Smagt. *An Introduction to Neural Networks* (8<sup>th</sup> Edition). The University of Amsterdam, 1996.
- [Gonz02] Gonzales, R. and R. Woods. *Digital Image Processing* (2<sup>nd</sup> Edition). Upper Saddle River, NJ: Prentice Hall, 2002.
- [Lisb92] Lisboa, P. *Neural Networks Current Applications*. London: Chapman & Hall, 1992.
- [Lite96] Liter, J. and H. Bulthoff. "An Introduction to Object Recognition," Max-Planck-Institut fur Biologische Kybernetik, Technical Report 43, 1996.
- [Ma04] Ma, Y. and others. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer: New York (2004).
- [Matl02] Math Works. *MATLAB User's Guide*. 2002.
- [Mend01] Mendenhall, M. *Wavelet-Based Audio Embedding and Audio/Video Compression*. MS thesis, AFIT/GE/ENG/01M-18, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 2001.
- [Mins67] Minsky, M. *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ: Prentice-Hall, 1967.
- [Oliv01] Oliveira, L. and others. "High-level Verification of Handwritten Numeral Strings," *Proceedings of SIBGRAPI*, 2001: 36-43 (2001).
- [Omid99] Omidvar, O., W. Cheung, T. Tambouratzis, S. Li. *Progress in Neural Networks – Volume 6 Shape Recognition*. Portland, OR: Intellect Books, 1999.

[Poli04] Poliker, R. *The Wavelet Tutorial*. Iowa State University, 1996. Accessed on 18 Jan 2004 <http://engineering.rowan.edu/~polikar/WAVELETS>.

[Sar197] Sarle, W. "Neural Network FAQ," Usenet newsgroup comp.ai.neural-nets, (1997) URL: <ftp://ftp.sas.com/pub/neural/FAQ.html>

[Schn00] Schneiderman, H. and T. Kanade. "A Statistical Method for 3D Object Detection Applied to Faces and Cars," *IEEE 2000*, 1063-6919 (2000).

[SNNS] *SNNS: Stuttgart Neural Network Simulator User Manual*, Version 4.2. University of Tübingen. 10 Jan 2004 <http://www-ra.informatik.uni-tuebingen.de/SNNS/>

[Sont92] Sontag, E. "Feedback Stabilization Using Two Hidden-Layer Nets," *IEEE Transactions on Neural Networks*, 3(6):981-990 (Nov 1992).

[Tesa88] Tesauro, G. and R. Janssens. "Scaling Relationships in Back-Propagation Learning," *Complex Systems*, 2:39-44 (1988).

[Torr00] Torres, L., J. Ruiz, L. Sucar, and G. Gomez. "Translation, Rotation and Scale Invariant Object Recognition," *IEEE Transactions on SMC*, Part C 30(1):125-130 (Feb. 2000).

[Wats94] Watson, A. "Image Compression Using the Discrete Cosine Transform," *Mathematica Journal*, 4(1): 81-88 (1994).

[LeCu89] LeCun Y., L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard. "Handwritten Digit Recognition: Applications of Neural Net Chips and Automatic Learning," *IEEE Communication*, 11:41-46 (1989).

## **Vita**

Captain Matthew Eyster served in the United States Army for four years before attending New Mexico State University. He graduated in 1999 with a B.S. in Electrical Engineering and was commissioned as a second lieutenant in the United States Air Force. He was first assigned to the National Air Intelligence Center (NAIC) at Wright Patterson Air Force Base, Ohio where he served as an Imagery Systems and Computer Engineer. After his assignment at NAIC, he was selected to attend the Air Force Institute of Technology, also located at Wright Patterson Air Force Base, Ohio. Upon graduation, Captain Eyster will be assigned to the Munitions Directorate, Air Force Research Laboratory, Eglin Air Force Base, Florida.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 23-03-2004		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) March 2003 – March 2004	
4. TITLE AND SUBTITLE  DISCOVERING THE MERIT OF THE WAVELET TRANSFORM FOR OBJECT CLASSIFICATION				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Eyster, Matthew D., Captain, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB, OH 45433-8865				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GE/ENG/04-09	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFTA Attn: Dr Robert Ewing Avionics Circle WPAFB, OH 45433 DSN: 785-6653x3592 e-mail: Robert.Ewing@wpafb.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT  APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
<p><b>14. ABSTRACT</b> Vision is the primary sense by which most biological systems collect information about their environment. Computer vision is a branch of artificial intelligence concerned with endowing machines with the ability to understand images. Object recognition is a key part of machine vision with far reaching benefits ranging from target recognition, surveillance systems, to automation systems.</p> <p>Extraction of salient features from an image is one of the key steps in object recognition. Typically, geometric primitives are extracted from an image using local analysis. However, the wavelet transform provides a global approach with good locality. Additionally, the directional and multiresolution properties may be exploited as a pre-processor to a neural network.</p> <p>This thesis examines the benefits of the wavelet transform as a pre-processor to a neural network for object recognition. Scaling of the wavelet coefficients and different neural network topologies are investigated. The system developed in this research is not intended to be critiqued on its classification performance. It only successfully classifies about 20% of the photographed models, however more important is the determination of the benefits of the wavelet transform, the effects of the various post-wavelet scaling functions, and the best neural network topology for this research. This is done by analyzing the system's performance on CAD models.</p>					
15. SUBJECT TERMS Artificial intelligence; identification systems; image processing; feature extraction; images – matching; three dimensional; neural nets; wavelet transforms; target classification					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Michael L. Talbert, Lt Col, USAF
U	U	U	UU	156	19b. TELEPHONE NUMBER (Include area code) (937) 255-3636, ext 4716 (Michael.Talbert@afit.edu)

